Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5521-95-9781

Ordinal Optimization of Admission Control in Wireless Multihop Integrated Networks via Standard Clock Simulation

JEFFREY E. WIESELTHIER
CRAIG M. BARNHART

Communication Systems Branch
Information Technology Division

ANTHONY EPHREMIDES

Locus, Inc.
Alexandria, Virginia

and

University of Maryland
College Park, Maryland

August 11, 1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 11, 1995	

4. TITLE AND SUBTITLE

Ordinal Optimization of Admission Control in Wireless Multihop Integrated Networks via Standard Clock Simulation

5. FUNDING NUMBERS

PE - 61153N
PR - RR015-09-41
WU - DN159-036

6. AUTHOR(S)

J.E. Wieselthier, C.M. Barnhart, and A. Ephremides*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/5521--95-9781

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*Locus, Inc., Alexandria, Virginia
University of Maryland, College Park, Maryland

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

In this report we apply the ideas of ordinal optimization and the technique of Standard Clock (SC) simulation to the voice-call admission-control problem in integrated voice/data multihop radio networks. We first describe the use of the SC approach on sequential machines, and quantify the speedup in simulation time that is achieved by its use in a number of queueing examples. We then develop an efficient simulation model for wireless integrated networks based on the use of the SC approach, which permits the rapid parallel simulation of a large number of admission-control policies. We have extended the basic SC approach by incorporating fixed-length data packets, whereas SC simulation is normally limited to systems with exponential interevent times. Using this model, we demonstrate the effectiveness of ordinal-optimization techniques, which provide a remarkable good ranking of admission-control policies after relatively short simulation runs, thereby facilitating the rapid determination of good policies. Moreover, we demonstrate that the use of crude, inaccurate analytical and simulation models can provide highly accurate policy rankings that can be used in conjunction with ordinal-optimization methods, provided that they incorporate the key aspects of system operation.

14. SUBJECT TERMS

Communications network
Admission control
Ordinal optimization
Standard clock
Voice/data/integration
Discrete event dynamic systems
Simulation

15. NUMBER OF PAGES

49

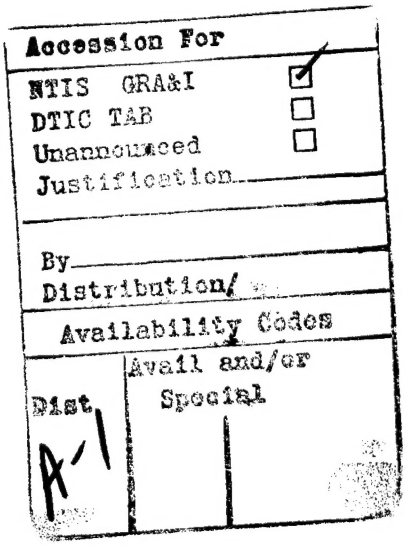
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS



ORDINAL OPTIMIZATION OF ADMISSION CONTROL IN WIRELESS MULTIHOP INTEGRATED NETWORKS VIA STANDARD CLOCK SIMULATION

1. INTRODUCTION

Simulation is the primary method for the performance evaluation of many forms of discrete-event dynamic systems (DEDS) for which analytical models have not yet been developed. Although in many cases simulation can provide a good estimate of system performance, it can also be extremely time consuming, and therefore expensive. This is especially true when it is necessary to evaluate system performance for a large number of values of one or more parameters or for different control policies.

Recent research in the area of DEDS has resulted in the development of a number of approaches that greatly improve the efficiency of the simulation process. For example, the Standard Clock (SC) technique developed by Vakili et al. [1-3], under which a common set of events with exponentially distributed interarrival times is used to simulate many sample paths in parallel, works well in problems involving either discrete or continuous parameters. However, despite the speedup that can be achieved by means of SC simulation, the task of evaluating performance for a large number of policies remains difficult.

Recently, Ho et al. [4] proposed the use of "ordinal optimization" as an approach to find "good," although not necessarily optimal, solutions to problems involving DEDS. Here, *ordinal* refers to the determination of policies that perform relatively well compared to other candidate policies, without necessarily obtaining accurate estimates of the performance values associated with these policies.

In this report we evaluate the improved efficiency that is achieved by using the SC approach for the simulation of several examples of DEDS, and demonstrate the effectiveness of the combined use of SC and ordinal-optimization techniques. First, we describe the SC approach to simulation, and quantify the speedup that can be achieved in some queueing system examples. Although the SC approach is ideally suited to parallel computing, considerable improvement in efficiency can also be achieved on a sequential computer, as we demonstrate here. We identify and track the time spent in the various aspects of the simulation, including event generation and state updating, and we compare the predicted performance improvement with that which was actually measured. Although a number of papers have been written on the SC approach, they do not address in detail the amount of time spent in each of the components of the simulation.

We then turn our attention to the admission-control problem in circuit-switched voice-only, multihop, wireless networks. The objective is to determine the optimal coordinate-convex [5] admission-control policy, i.e., the policy that optimizes an appropriate performance criterion such as voice-call blocking probability. The use of such policies, when voice calls are described by Poisson arrival statistics and exponentially distributed call durations, leads to a product-form characterization of the probability distribution of the voice-call system state, which in turn permits the straightforward evaluation of system performance. However, the required calculations are computationally intensive, which creates a situation

Manuscript approved April 12, 1995

further aggravated by the need to compute performance for a large number of policies as part of the search for the optimal one. We demonstrate how the SC approach can be applied to this problem, and we evaluate the improvement in simulation efficiency that can be achieved.

The primary focus of our report, however, is the modeling of integrated voice/data networks, which are a generalization of the voice-only networks just described. The study of integrated networks is of considerable interest to the communication network community. We demonstrate how the SC model developed for voice-only operation can be generalized to incorporate fixed-length data packets; this is a significant extension of the SC methodology, which is normally applicable only to systems with exponential interevent times. The performance metrics evaluated by simulation are voice-call blocking probability and data-packet delay. However, our focus is primarily on the ordinal rankings of the policies, rather than on the accurate evaluation of these performance measures.

A crucial observation from our ordinal-optimization studies, and one that has interesting and possibly far-reaching implications in the study of optimization methods, is that crude models are often adequate to predict the relative performance of different control policies. For example, we demonstrate that exceptionally accurate ordinal policy rankings can be obtained from simulation models that incorporate key aspects of the systems being modeled, although they do not incorporate all aspects of system operation and do not predict performance (e.g., delay) accurately. Furthermore, we have also obtained highly accurate policy rankings using a crude analytical model for delay performance. This suggests that simple analytical models can be used to reduce the search space to just a few policies (perhaps to just a single one) whose performance can then be evaluated accurately via simulation, thus decreasing computation time dramatically. Although further research is needed to verify the effectiveness of ordinal-optimization methods in more-general scenarios, the results obtained thus far suggest that ordinal-optimization techniques based on simple analytical models would be easy to implement in practical communication systems, and that they would provide satisfactory performance.

1.1 Outline of the Report

In Section 2 we use the M/M/1/K queue paradigm to explain the SC methodology, and in Section 3 we discuss SC simulation timing results for M/M/1/K queues. These results demonstrate that the SC approach provides considerable improvement in simulation efficiency as compared with conventional brute-force techniques. In Section 4 we extend the use of SC simulation to networks of queues, and in so doing demonstrate that the SC method scales well with problem complexity.

In Section 5 we switch our focus to integrated voice/data networks, which are the primary subject of this report. We define the admission-control problem, first for voice-only networks and then for integrated networks. In Section 6 we discuss the application of SC techniques to voice-only networks, and in Section 7 we demonstrate how they can be extended to integrated networks. Our studies of simulation timing results for integrated networks have discovered previously undocumented behavior related to memory considerations, which limits the speedup that can be achieved on sequential machines; however, we have developed a modification to the SC method that mitigates such effects almost completely.

In Section 8 we review the principles of ordinal optimization. Then in Sections 9 and 10 we present our simulation performance results, which demonstrate the effectiveness of ordinal optimization as a technique for the determination of good control policies without necessarily obtaining a good estimate of system performance. In Section 11 we make some observations on why ordinal optimization is an effective tool, and finally, in Section 12, we present our conclusions from this study.

2. THE STANDARD CLOCK APPROACH TO SIMULATION

We begin by reviewing the principles of SC simulation [1-3], which permits the simultaneous evaluation of system performance under a large number of control policies. In this report we limit our discussion to the case of sequential machines; SC simulation on parallel machines will be the subject of a future report. We use the M/M/1/K queue paradigm to explain the SC methodology, and in later sections we apply SC techniques to circuit-switched voice networks, and finally to the integrated voice/data networks that are the focus of this report.

We consider an M/M/1/K queue, i.e., a single-server queue with finite buffer capacity K (including the packet in service), Poisson arrival process at rate λ, and exponentially distributed service of expected duration $1/\mu$. We use the following notation to describe the state of the queueing system:

x = the number of packets in the system,
$\pi(x)$ = the steady-state probability that there are x packets in the system.

The discrete parameter K in the M/M/1/K queue makes this system a particularly good candidate for SC simulation, which works equally well for continuous or discrete variables. Furthermore, the availability of an analytical model for the M/M/1/K queue [6] makes it a good problem for the development and illustration of simulation models, since it permits verification of the accuracy of the simulations. In particular, most of our simulation results are for the case of $\rho = \lambda/\mu = 1$, for which it is easy to show that the queue length distribution is uniform, i.e., $\pi(x) = 1/(1+K)$, $x = 0, 1, \ldots, K$.

Typically, in traditional simulations of M/M/1/K queues, an event calendar is used to schedule events (i.e., arrivals and departures) that are generated by using a distinct distribution for each event type. By contrast, in SC simulations, a single sequence of random numbers is generated from an exponential distribution with parameter $\Lambda = \lambda + \mu$. Each of the variates in this sequence represents a generic interevent time. The type of event associated with each interevent time is determined by drawing an independent random number U from a uniform distribution on the interval [0,1]. The event is an arrival if $U \leq \lambda/\Lambda$, otherwise it is a departure. Generation of events at this rate Λ, which is referred to as the *maximal rate*, is an example of the well-known technique of *uniformization* [7]. A "ratio yardstick" showing the event-type determination process is shown in Fig. 1.[1]

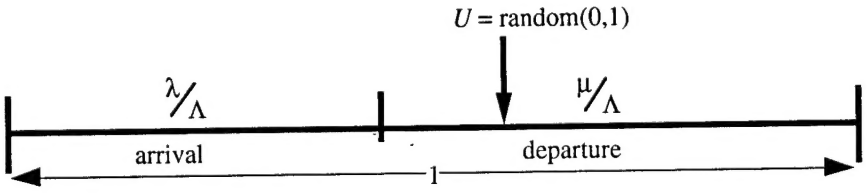


Fig. 1 – Ratio yardstick for the M/M/1/K simulations

Thus two random numbers must be generated to produce each event; one to specify the timing of the next event, and the other to specify its type. It is possible that an event determined in this manner turns out to be infeasible (e.g., a departure from an empty system). The interevent time of such a

[1] The ratio yardstick is easily extended to complex examples with many different types of events. As in the present example, events are generated at the maximal rate (which is the sum of all exponential interevent rates in the system), and each event corresponds to a region on the yardstick, the width of which is equal to the probability of the corresponding event. The event type is determined by the region into which a random number (drawn from a uniform distribution on [0,1]) falls. The computational effort involved in determining the event type by means of the alias method [8] is independent of the number of event types, thus making this method suitable for complex systems.

"fictitious" event is used to update the system time as if the event were "real" and did in fact occur, but no state change occurs (the fictitious event is discarded).

The improved efficiency of the SC method is achieved by using the resulting sequence of (interevent time, event type) pairs, known as the *clock sequence*, to simultaneously generate sample paths for a number of structurally similar, but parametrically different, systems. In particular, a single clock sequence can be used to generate N sample paths in parallel for N M/M/1/K queues, each with a different value of K.

Figure 2 shows a simplified comparison of SC and conventional, or *Brute-Force* (BF), simulation for the case of 100 sample paths ($N = 100$) and a simulation duration of 1000 events, where a sequential machine is used in both cases. The program structure is shown as nested "Do loops." In BF simulation the "Do 100 sample paths" is the outer loop, and within this loop the "Do 1,000 events" loop actually constructs each of the sample paths. With this loop structure, 100,000 events and 100,000 state updates are required. In contrast, with SC simulation the order of the "Do loops" is reversed. As a consequence of this reordering, the "Generate Event" procedure is outside the inner loop, and each event generated is passed to all 100 sample paths. Thus, with SC simulation only 1,000 events need to be generated, although we still have to perform the 100,000 state updates as in the BF case.[2]

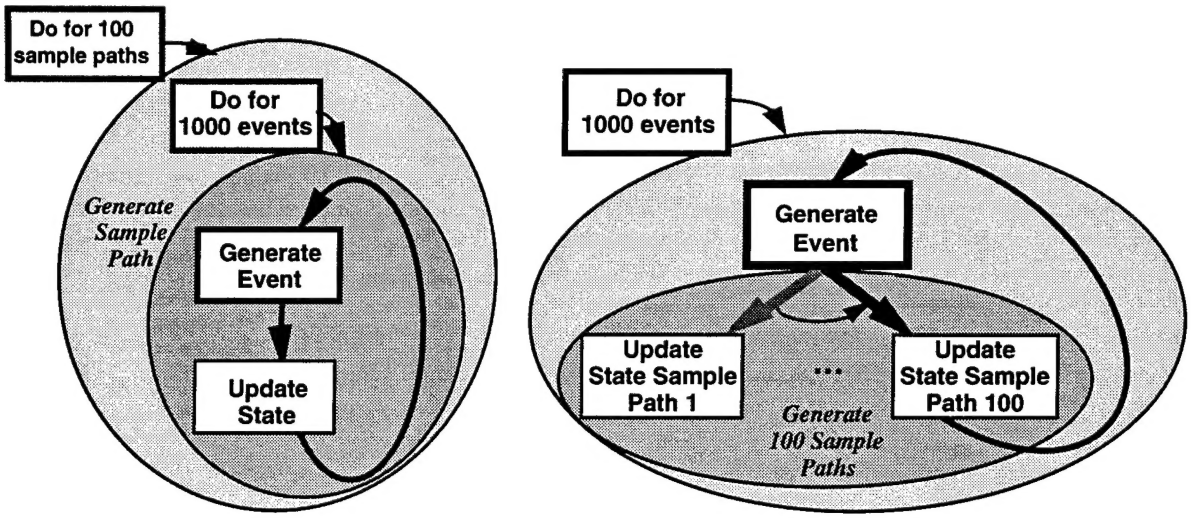


Fig. 2 – A comparison of program structure in BF and SC simulations

In the M/M/1/K example, arrival events are always feasible for all values of K; however, an arrival when the buffer is full, i.e., when $x = K$, does not lead to a change of state because it is blocked. Similarly, a fictitious event, e.g., a departure when $x = 0$ in this system, leaves the state unchanged. Thus a different trajectory may be produced in each of these experiments, even though the clock sequence is the same for all. Since each element of the clock sequence is used N times, the number of events that must be generated is reduced by a factor of N. This reduction has a dramatic effect on the overall

[2] The SC technique is extremely well suited for implementation on parallel machines such as the Connection Machine CM-5. On such a machine, the front end generates the events, each of which is passed to all of the parallel processors, each of which is responsible for the updating of one or more sample paths. Although the SC speedup on parallel machines can be much greater than that achievable on sequential machines, in this report we limit our discussion to sequential machines. One of our principal results is that significant speedup can be achieved by using SC techniques (as compared to BF techniques) on a sequential machine. SC simulations on parallel machines will be the subject of a future NRL report.

simulation time because the generation of events is considerably more time consuming than the consequent updating of system state, as our simulation timing results show in Sections 3 and 4.

Unfortunately, the speedup achieved by using the same event sequence for all N sample paths can be reduced by the occurrence of fictitious events; thus, additional events must be generated to observe the specified number of real events. However, in Section 3 we demonstrate that significant speedup can be obtained even when the fraction of fictitious events is high. Another limitation of this approach is that it is normally restricted to systems with exponentially distributed interarrival times, although it has recently been shown that some deterministic events [9] or other nonexponentially distributed events [10] can also be incorporated into the model and, in fact, we demonstrate in Section 7 how fixed-length data packets can be incorporated into SC simulation models.

3. SC SIMULATION TIMING RESULTS FOR M/M/1/K QUEUES

To measure the increase in simulation efficiency that can be obtained by using the SC method, we compare it to BF simulation. Instead of performing a true BF simulation in which separate event streams would be generated for each event type, we actually performed a separate SC simulation for each value of K. Thus our BF simulations consisted of N SC simulations (each for a single value of K) performed sequentially. Therefore, in our simulations of M/M/1/K queues, both the SC and the BF runs must send approximately the same number of events to each sample path. Our use of the SC approach to model BF simulations results in somewhat reduced efficiency, because *true* BF simulations do not generate fictitious events, whereas SC simulations do. However, the SC approach eliminates the need to maintain and check a state-dependent feasible event set that would normally be needed in BF simulations. Furthermore, since the number of fictitious events in our simulations (for $\rho = 1$) is quite small (and decreases as K increases), their generation does not significantly penalize our timing measurements for the BF approach. The timing error introduced by our use of this approach in BF simulation is, in fact, insignificant in comparison with the degree of improvement that is achieved by using the SC simulation method. A more-detailed description of our BF and SC simulation models is given in [11, 12].

The SC and the BF simulation models for M/M/1/K queues were programmed in C++ and run on a Sun-4/330 workstation, which is a sequential machine. Our focus here is on the speedup achieved by using the SC approach as compared to BF simulation, rather than on the statistics of the simulated processes (buffer occupancies, etc.).

Figure 3 compares the time required to simulate N sample paths using BF and SC methods for the case of $\rho = 1$. Similar timing results for SC and BF simulations of M/M/1/K queues for values of K from 3 to 10 were presented in [13]. Besides studying much larger values of K, our results differ in that we have also identified and tracked the time spent in various aspects of the simulation. Our detailed examination of simulation timing results has permitted the development of an estimate of the improvement in simulation time that can be achieved using the SC approach on a sequential machine.

The timing results shown in Fig. 3 are based on averages taken over 10 runs using different random number generator seeds. The SC simulations were terminated when the queue with $K = 2$ had received 100,000 real events. We have observed that an average of 119,967.1 events (both real and fictitious) were required for these simulations. The BF simulations of each sample path were terminated after 119,967 events (real or fictitious).[3]

[3] Since the N sample paths are not generated in parallel in the BF simulations, and because the sequence of random variates is different for each sample path, we do not know a priori how many events are required for a queue with $K = 2$ to see 100,000 real events. Therefore we use the average number of events (determined from 10 different SC simulations with different random number generator seeds) required for termination, namely 119,967.

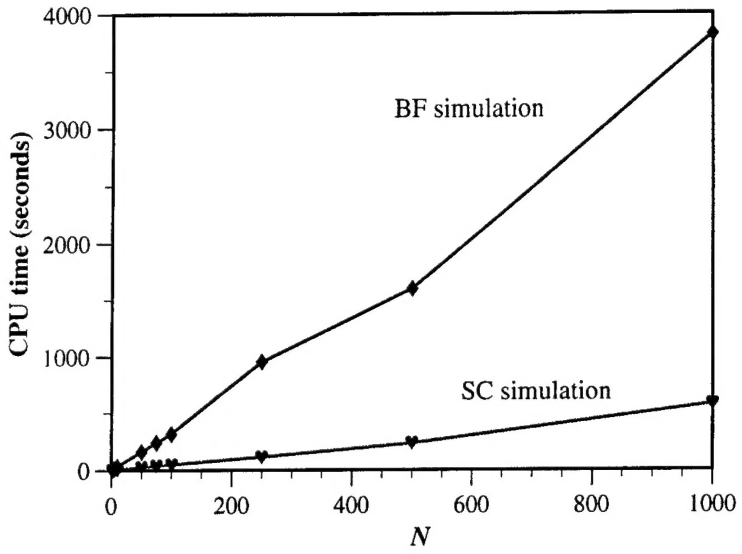


Fig. 3 – The time required for SC and BF simulations of N M/M/1/K queues, $K = 1, ..., N$

Figure 3 shows that a significant speedup (a factor of 6.62 for $N = 1000$) is achieved by using SC simulation as opposed to BF simulation, and that in both cases the total simulation time is roughly linear in N. The primary source of this reduction in simulation time is the efficient use that the SC simulation makes of each event that is generated. By generating the sample paths in parallel, every event generated is used in each of the N sample paths.

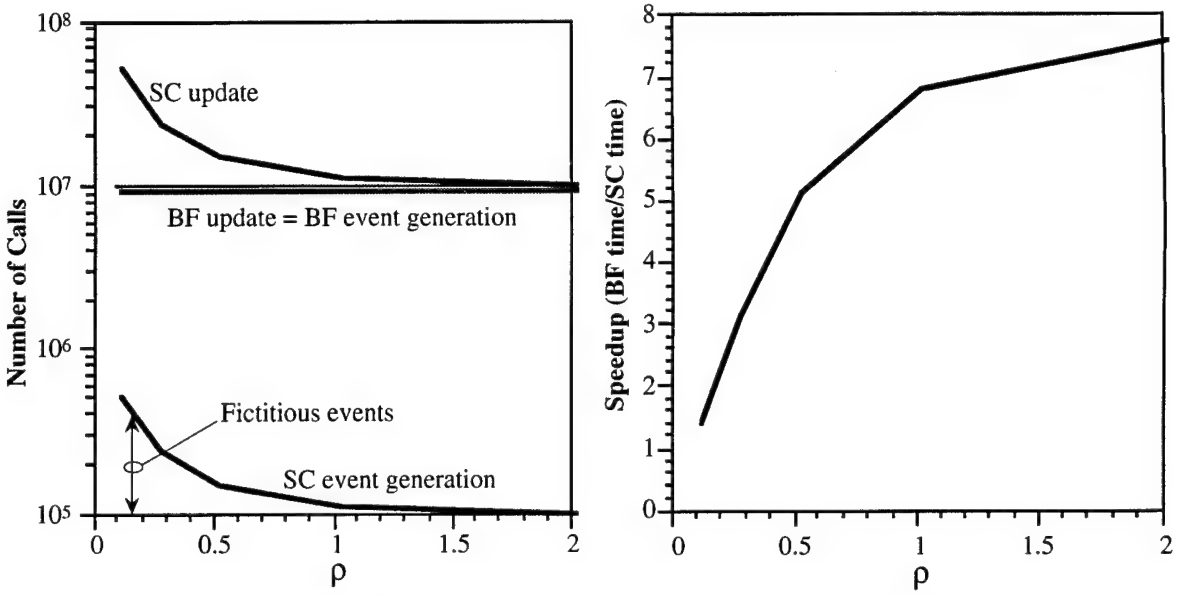
The degree of improvement in efficiency that can be achieved depends on the efficiency with which each of the generated events can be used, which in turn depends on the number of fictitious events that are generated. As noted above, for $\rho = 1$ the number of events that had to be generated was approximately 120% of the number of real events that were desired. As ρ is decreased below 1, the number of fictitious events increases, and thus the efficiency of the simulation decreases.

Figure 4(a) shows the impact of ρ on the number of events that must be generated in simulations of the M/M/1/K queue. In a BF simulation there are no fictitious events; thus to perform 100 parallel simulations, each consisting of 10^5 events (total of arrivals and departures), 10^7 events must be generated and 10^7 state updates must be performed. In the SC simulations, events were generated until the sample path for $K = 2$ received 10^5 real events (and therefore the sample paths for $K > 2$ received at least that number). The number of fictitious events (i.e., fictitious for $K = 2$, although possibly real for larger values of K) increases as ρ decreases, resulting in the need to generate additional events; the number of state updates is 100 times the number of events. Figure 4(b), which shows the resulting speedup as a function of ρ, demonstrates that the speedup is greater than 1 even when 80% of the events are fictitious (the case of $\rho = 0.1$).

By using the Sun performance analysis tool "gprof" [14], we were able to break down the time spent in each of the major simulation functions, namely the *state update function* and the *event generation process*.[4] The results for BF and SC simulations of 100 M/M/1/K queues (i.e., $K = 1, 2, ..., 100$) with $\rho = 1.0$ are shown in Table 1. The BF results are from a single simulation of the queues; the SC results are the average taken over 10 simulations (with different random number generator seeds) of the queues. The table shows the number of calls to each function, the total time in seconds spent to satisfy these calls, the

[4] The event generation process requires the generation of two random numbers (uniform variates), one log operation to transform a uniform variate into an exponentially distributed one, and an operation that maps a uniform variate to an event type.

percentage of the total simulation time that was spent in each function, and the average time per function call.[5] The first row represents the time expense involved in actually manipulating a sample path, i.e., state updating and statistics collection. The bottom row of the table shows the total event generation cost.



(a) Events and state updates (b) Simulation speedup

Fig. 4 – Variation of the number of events and state updates, and the corresponding simulation speedup, as a function of ρ for the M/M/1/K system (100 sample paths)

Table 1 – Timing Breakdown for BF and SC Simulations of 100 M/M/1/K Queues.

	BF simulation, $N = 100$, ρ=1				SC simulations, $N = 100$, ρ=1			
function	calls	time (s)	%time	time/call	calls	time (s)	%time	time/call
update	11,996,700	52.90	21.0	4.410μs	11,996,710	54.24	96.6	4.521μs
event generation	11,996,700	199.06	79.0	16.592μs	119,967.1	1.93	3.4	16.105μs

We define the simulation speedup to be

$$\text{speedup} = \frac{\text{BF simulation time}}{\text{SC simulation time}},$$

which in this example is 4.485. Focusing specifically on the time spent generating events, the speedup is (199.06/1.933=) 102.98, which is slightly better than the expected value of 100.

For the general case of simulating N sample paths, we estimate the *per-event* simulation speedup, denoted by $S_e(N)$, as

[5] The update function and the event generation process are identical in both the BF and SC programs. Although the times measured per function call are not identical, they are sufficiently close to verify the accuracy of our timing estimates.

$$S_e(N) = \text{speedup} \cdot \frac{\text{number of SC events/sample path}}{\text{number of BF events/sample path}} = \frac{\text{BF simulation time/event}}{\text{SC simulation time/event}}$$
$$= \frac{N\big(t(\text{path update}) + t(\text{event generation})\big)}{N \cdot t(\text{path update}) + t(\text{event generation})},$$

where $t(\bullet)$ is the time per function call. Because our BF simulation is simply a SC simulation in which the individual sample paths are generated in sequence rather than in parallel, both the SC and the BF runs must send approximately the same number of events to each sample path. In general however, because fictitious events need not be generated in the BF approach, a BF simulation can generate a sample path of given length with fewer events than a SC simulation of the same model. Thus, the *per-event* simulation speedup estimate $S_e(N)$ accommodates differences in the number of events required in BF and SC simulations.

As N increases, the per-event speedup achievable by using SC simulation on a sequential machine approaches the following limit

$$\lim_{N\to\infty} S_e(N) = 1 + \frac{t(\text{event generation})}{t(\text{path update})}.$$

This equation shows that, on a per-event basis for large N, the speedup achieved with SC simulation on a sequential machine is controlled by the ratio of t(event generation) to t(path update), and that SC always provides a per-event speedup value greater than 1.

Figure 5 compares the speedup found in simulations to our estimated per-event speedup $S_e(N)$. It shows that the speedup obtained in simulations is actually greater than that of our estimate. Although there is a discrepancy, both the predicted speedup and that found by simulation confirm that there is an asymptotic value that the SC simulation speedup can achieve on a sequential machine, i.e., that the speedup does not continue to increase significantly once N has reached a sufficiently high value.

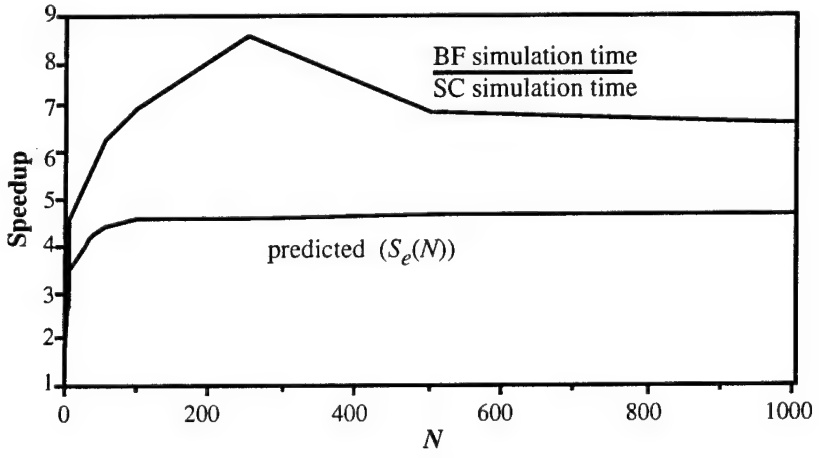


Fig. 5 – Actual and predicted speedup in the simulation of N M/M/1/K queues, K = 1, ..., N

It is difficult to estimate precisely the timing in simulation runs, in part because the measurement tools disturb the quantities that they are measuring, and because of a number of subtle platform-dependent issues such as memory management and caching [15]. The recognition of these issues provides another potential path for performance optimization. We explore this path in Section 7.3, where we discuss the speedup achieved in SC simulations of integrated networks. Additionally, our estimate accounts only for

the two major functions; it neglects time spent in other portions of the simulation such as initialization and the "main" driver portion of the program. The simulation timing results include all aspects of the simulation, and are based on runs in which the measurement tools are not used. Thus they provide a more accurate measure of the speedup provided by the SC method on a particular platform, while our estimate gives a general approximation of the speedup, independently of platform variations.

4. SC SIMULATION OF NETWORKS OF QUEUES

We extend the use of SC simulation to the examination of networks of queues that in isolation would be M/M/1/K. Of course, once the M/M/1/K queues are interconnected, they cease to be M/M/1/K in that the arrival process at each of them is no longer Poisson (since the outputs of the queues are no longer independent).

4.1 The Network Model

Consider the network of four fully interconnected queues as shown in Fig. 6. Queue i receives exogenous Poisson arrivals at rate r_i, has buffer capacity K_i, and it routes a departing packet to queue j with probability P_{ij},[6] or the packet exits the network with probability P_{ie}. Thus, along with its exogenous input, each queue receives a fraction, depending on the P_{ij}, of the (non-Poisson) output from the other queues (three, in this example); the total input to queue i is

$$\lambda_i = r_i + \sum_{j=1}^{J} \lambda_j \left(1 - \pi\left(K_j\right)\right) P_{ji}, \quad 1 \le i \le J,$$

where the factor $(1-\pi(K_j))$ accounts for the traffic thinning that occurs at each queue as a result of the finite buffer capacity.

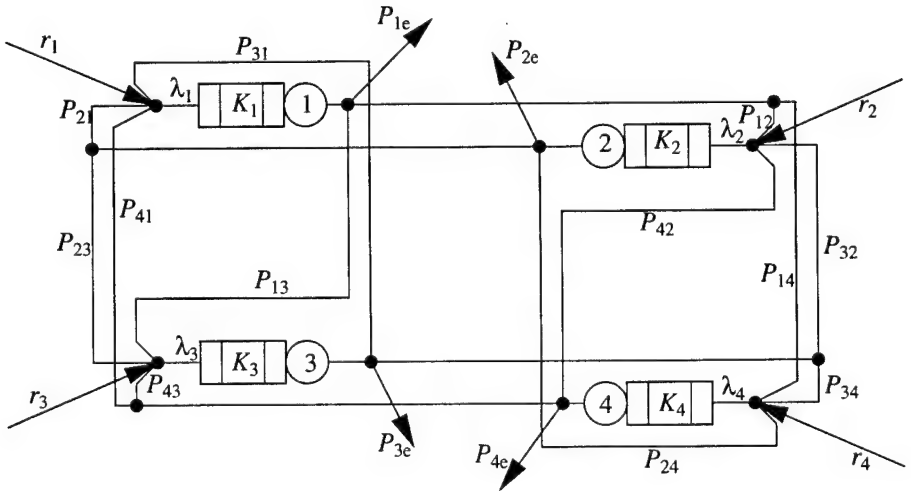


Fig. 6 – A four-queue network

Standard Clock simulation of a network of queues proceeds in nearly the same manner as for a single queue; however, the class of events that must be generated is richer. We use the notation a_i to denote an exogenous arrival for queue i, t_{ij} to denote a "transfer" event from queue i to queue j, and t_{ie} to indicate that the departure from queue i is exiting the network. A ratio yardstick showing the set of all events for the network of Fig. 6, along with the probability of their occurrence, is shown in Fig. 7.

---

[6] Although not strictly necessary, we assume here that $P_{ii} = 0$.

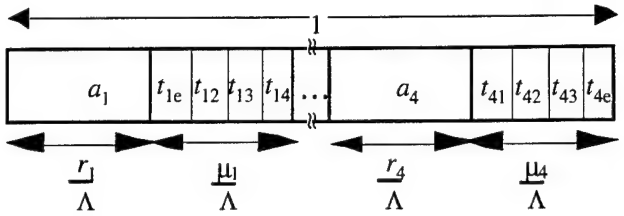


Fig. 7 – Ratio yardstick showing the set of events for the network of Fig. 6

As is implied by Fig. 7, the maximal event rate Λ for our example four-queue network is the sum of the exogenous arrival rates plus the sum of the server rates:

$$\Lambda = \sum_{j=1}^{4} \left(r_j + \mu_j \right).$$

Thus, an event is an exogenous arrival to queue i with probability r_i/Λ. Similarly, it is a departure from queue i with probability μ_i/Λ. As shown in Fig. 7, this probability quantity is subdivided into the four possible transfer events, i.e., the departure event is a transfer from queue i to queue j with probability

$$P\left(t_{ij}\right) = \begin{cases} \dfrac{P_{ij}\mu_i}{\Lambda} & i \neq j \\ \dfrac{P_{ie}\mu_i}{\Lambda} & j = \mathrm{e} \end{cases}.$$

Notice that the event generation process is independent of the buffer capacities. The fact that the queues have finite buffers affects the simulation only in the state update function.

In Table 2, we show a timing breakdown from a run that simulated the network of Fig. 6 with 100 different parameter settings: $K_1 = 1, \ldots, 100$; $K_2 = K_3 = K_4 = 100$. The remaining parameters were: $r_i = \mu_i = 0.2$, $i = 1, \ldots, 4$; $P_{ij} = P_{ie} = 0.25$, $i,j = 1, \ldots, 4$, $i \neq j$. As expected, the time required per call to the sample-path update function (approximately 5.936 μs) is larger than that required for the M/M/1/K queue model (measured at 4.521μs). However, we see that approximately the same percentage of simulation time is spent actually updating the sample paths in both the network model and the single queue model.

Table 2 – Timing Breakdown for SC Simulations of Networks of Queues.

	SC simulations; $N = 100$			
function	calls	time (s)	%time	time/call
update	10,000,000	59.36	97.4	5.936μs
event generation	100,000	1.59	2.6	15.890μs

These results indicate that the SC method scales well with problem complexity. There are four times the number of queues and 10 times as many events to consider in this network simulation, and the update function is more complex than that for the M/M/1/K queue. However, the time per call to the update function is only 1/3 longer for the network simulation. A comparison of the event-generation time in Tables 1 and 2 indicates a high level of consistency between our simulations of the two systems. This is not unexpected since the random number generation operation and log are clearly identical in both systems, and, as mentioned previously, the use of the alias method makes the computational effort involved in determining the event type independent of the number of event types.

5. THE ADMISSION-CONTROL PROBLEM IN INTEGRATED VOICE/DATA NETWORKS

First we describe the basic circuit-switched model used for voice traffic, and then we extend that model to incorporate integrated voice/data operation.

5.1 The Circuit-Switched Voice Network Model

We consider multihop radio networks in which voice-call service is provided by the establishment of a circuit over a predetermined path between the originator of a call and its destination node for the duration of the call. We assume the use of frequency-division multiple access (FDMA) with a sufficient number of frequencies to provide contention-free channel access; thus a necessary and sufficient condition for a call to be established is that a transceiver is available[7] at every node in its circuit. The model is described in greater detail in [16] and [17].

Figure 8 shows an example of a ten-node wireless network that supports circuit-switched voice. The multihop paths corresponding to each of $J = 5$ circuits are distinguished by different shadings. The dashed lines connect nodes that are within communication range of each other, but that do not support any of the five circuits.

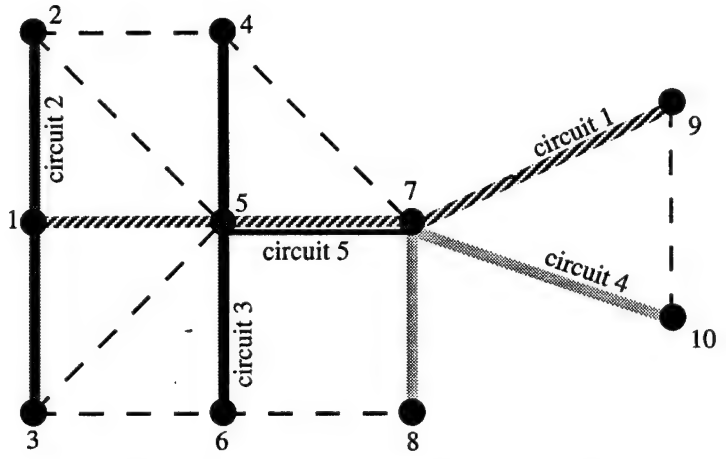


Fig. 8 – An example circuit-switched voice network

Calls for circuit j are generated according to a Poisson process with rate λ_j^V, and their durations are exponentially distributed with parameter μ_j^V. A vector description of circuit j in terms of the nodes it traverses is given by $c_j = \{c_{j1}, c_{j2}, ..., c_{jN}\}$, where

$$c_{ji} = \begin{cases} 1, & \text{if circuit } j \text{ traverses node } i \\ 0, & \text{otherwise} \end{cases},$$

and N is the number of nodes in the network. The state of the system with J source-destination pairs, and hence J circuits, is described by the vector $\boldsymbol{x} = \{x_1, x_2, \ldots, x_J\}$, where x_j is the number of calls currently active on circuit j, each of which is referred to as a call of type j. A central controller makes the decisions on whether or not to accept calls based on perfect knowledge of the number of calls of each type that are currently in progress (i.e., the system state $\boldsymbol{x}$), and hence the set of resources that are available for new calls. The transceivers needed to establish a circuit are acquired simultaneously when the call arrives, and

[7] For a transceiver to be available, it must be idle. We do not consider preemptive policies in which a high priority call can preempt an ongoing call of lower priority.

are released simultaneously when the call is completed. Calls are blocked when one or more nodes along the path do not have a transceiver available, or when a decision is made not to accept a call despite the availability of transceivers. Blocked calls are lost from the system. These assumptions, coupled with the class of coordinate-convex admission-control policies discussed below, lead to a mathematically tractable description of system performance.

The number of transceivers at node i is denoted by T_i. No more than T_i calls can simultaneously use the resources at node i, i.e.,

$$\sum_{j=1}^{J} x_j c_{ji} \leq T_i, \qquad i = 1,\ldots,N.$$

These equations, which are termed the "system constraints," limit the state space Ω_0 in which $\boldsymbol{x}$ is allowed to take values. We refer to a system that is solely constrained by these system constraints as an "uncontrolled" system.

In some cases, network performance (e.g., blocking probability or throughput) can be improved by administering an admission-control policy that blocks some calls even though resources are available [17]. The admission-control policy is defined by a set of linear inequality constraints that effectively further restrict the admissible state space. For notational purposes, we subdivide the control policy, denoted by Ω, into a set of circuit "thresholds," and a set of "linear-combination constraints." Thresholds restrict the number of calls that will be admitted to the individual circuits, and can be expressed as

$$x_j \leq X_j = \text{threshold on circuit } j, \qquad j = 1,\ldots,J.$$

The linear-combination constraints, which are based on the fact that several circuits may pass through a particular node, place limits on the sum of the number of calls of several types, and are defined as

$$\sum_{j \in S_I} x_j \leq Y_I = \text{threshold on total number of calls of types that are members of set } S_I$$

for suitable subsets S_I of the set of all call types. In other words, we consider policies that, for each node, examine subsets S_I of the call types that pass through that node; the policy limits the total number of calls of types that are members of set S_I to the value Y_I. Thus, the control policy is given by $\Omega = \{X_1, \ldots, X_J, Y_1, \ldots\}$. Now the problem is the determination of the optimal admission-control policy Ω^*, i.e., the values of the X_j and the Y_I that yield the optimal network performance. Table 3 enumerates the linear-combination constraints of the form Y_I for the network of Fig. 8.[8]

The use of this form of control policy assures us that the state space is "coordinate convex" [5].[9] A coordinate-convex policy is specified in terms of the set of admissible states; completed calls are not blocked from leaving, on-going calls do not get rerouted, and new calls are admitted with probability 1 if the state to be entered is in the admissible region. It was demonstrated in [18] that coordinate-convex policies are optimal for "Multiple-Service Single-Resource" (MSSR) problems such as the sharing of a buffer by different classes of traffic. The voice-call admission problem under study here is not MSSR because it is characterized by multiple resources (namely the transceivers at each of the network nodes), and thus is a member of the family of "Multiple-Service Multiple-Resource" (MSMR) problems.

[8] Linear-combination constraints are not needed for nodes at which two or fewer circuits intersect [16, 17].

[9] It is also convex, although not all coordinate-convex policies correspond to convex state spaces.

Although coordinate-convex policies are not necessarily optimal for such problems, it was demonstrated in [19, 20] for several example problems that nearly optimal performance can be obtained through their use.

Table 3 — Linear-Combination Constraints for Network of Fig. 8.

Linear-Combination Constraint	Constraining Node
$x_1 + x_3 \le Y_1$	5
$x_1 + x_4 \le Y_2$	7
$x_1 + x_5 \le Y_3$	5
$x_3 + x_5 \le Y_4$	5
$x_4 + x_5 \le Y_5$	7

Under our assumption of Poisson arrival statistics, the use of coordinate-convex policies results in the so-called product-form characterization of the system[10] [19, 20], where the equilibrium state probability $\pi(\bullet)$ is given by

$$\pi(\boldsymbol{x}) = \pi(0)\prod_{j=1}^{J}\frac{\rho_j^{x_j}}{x_j!}, \qquad \boldsymbol{x} \in \Omega$$

where $\rho_j = \lambda_j^V/\mu_j^V$, and $\pi(0) = (\sum_{\boldsymbol{x}\in\Omega}\prod_{j=1}^{J}\rho_j^{x_j}/x_j!)^{-1}$ is the normalization constant. For any coordinate-convex policy, it is straightforward (though time consuming) to evaluate $\pi(0)$, which in turn permits the evaluation of performance measures such as throughput and blocking probability.

Thus, the goal is to restrict the admissible state space to a coordinate-convex region such that the desired performance measure is optimized. The direct approach is to compute $\pi(0)$ and the performance index for all possible coordinate-convex regions. In [16] and [17] we developed a recursive procedure based on the MSMR methodology introduced in [19, 20] to accelerate the evaluation of a large number of admission-control policies, and a descent-search method to reduce the number of policies that must be evaluated in searching for the optimal one. Despite the speedup that was achieved, the search remained computationally intensive. We demonstrate in Section 6 how the SC approach can be applied to the simulation of circuit-switched voice networks. But first we describe our model for integrated networks.

5.2 The Integrated Network Model

In the integrated network model, voice-traffic is modeled as in the voice-only network, i.e., Poisson arrivals at rate λ_j^V and exponential service with parameter μ_j^V on circuit j. Data traffic consists of fixed-length packets with Poisson arrival statistics. Both voice and data arrival events, as well as voice departure events, are stochastic. However, because the data packets are of fixed length, the packet service time and, hence, the data-packet departure events are deterministic, although they do depend on the voice state and data-packet arrival times.

---

[10] To determine the equilibrium state distribution $\pi(\bullet)$, the assumption of exponentially distributed call durations is not necessary. In circuit-switched networks with Poisson arrival statistics and coordinate-convex admission-control policies, the product-form solution applies for any call-length distribution, an example of the so-called insensitivity property [21]. Thus the equilibrium distribution depends only on the expected call duration, not on the distribution of call length. However, the statistics of the time spent in each state (which affect the expected data-packet delay in integrated networks) do depend on the call-length distribution.

At each node a separate queue is formed for packets intended for each of its neighbors. In particular, the arrival process at queue k is Poisson at rate λ_k^d. Figure 8(a) shows an integrated network in which one or more data-packet queues have been superimposed on several nodes of the network of Fig. 8. The system state is defined as $\{\boldsymbol{x}, \boldsymbol{d}\}$, where $\boldsymbol{x}$ is the voice state vector defined in Section 5.1, and $\boldsymbol{d}$ is a vector representing the lengths of the data-packet queues. At any time instant, the transceivers not being used to support voice traffic are available to support data traffic, which is conceptually handled in our simulation model as follows. Whenever one or more transceivers at a node are idle, the data queues at the node are examined. If any are occupied, and if a transceiver is also available at the destination node, the packet is transmitted; each data-packet transmission occupies one transceiver at both source and destination nodes for the duration of the transmission. We have also investigated an alternative model in which the availability of a transceiver is not required at the receiving node. This latter case implicitly assumes that receivers are abundant, and that only transmitters are a limited resource. We refer to these two models as "receivers verified" and "receivers assumed" respectively. In Section 10.2 we demonstrate that, although these models yield very different data-packet delay values, the ordinal rankings of voice-call admission-control policies (in terms of the resulting data-packet delay) are virtually identical for these two cases.

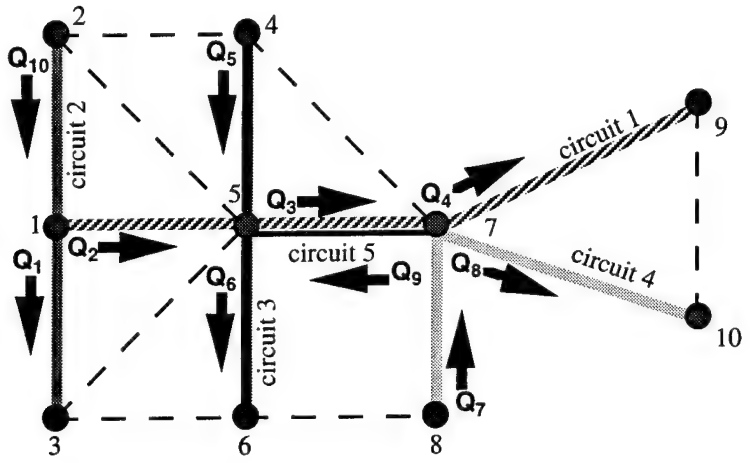


Fig. 8(a) – An example integrated voice and data wireless network

The decision to accept a voice call depends only on the number of voice calls (of each type) currently in progress in the system, and is independent of the size of data queues. Thus, voice traffic has priority over data traffic. Data packets are permitted to use any transceivers that are not occupied by voice, and are queued when resources are not immediately available. If a voice call arrives when data traffic is using one or more of the transceivers needed to support the call, the data packet is permitted to complete its transmission, after which the voice call begins. Since data packets are generally much shorter than voice calls, this should not interfere significantly with the assumption of exponential call durations.[11] These assumptions are consistent with those widely used in the "movable-boundary" scheme for integrated multiplexing [22], which has also been applied to channel access in wireless networks [23]. The integration scheme in the present report can be viewed as an implementation of the movable boundary in the "transceiver domain."

The simplest case, and the one we consider, is that of single-hop data traffic, i.e., all traffic generated at a node is to be delivered to one of its immediate neighbors (no relaying is required). Multihop systems may be modeled approximately by assuming a Poisson arrival stream of appropriate

[11] Alternatively, it could have been assumed that the data packet is preempted, necessitating its retransmission when the channel again becomes available.

rate at each queue along the multihop path, thus implicitly assuming that the queues at each node are independent; this is similar to Kleinrock's well-known independence assumption. The expected end-to-end delay would then be the sum of the expected delays at each hop along the path. Of course, this model is not exact, e.g., because the arrival processes at intermediate nodes are not Poisson and are not independent. However, reasonable approximations to the multihop path delays may be obtained from concatenated single-hop delays.

The network models studied here neglect the protocol issues associated with the coordination of transmitting and receiving platforms; as in [16] and [17], our focus here is on the performance that can be achieved in an ideal system in which full state information is available at all nodes, with no time delay or overhead cost. However, we would like to address briefly how data traffic might be handled in a practical system. One approach would be to establish a link-activation schedule that pairs neighboring nodes on a time-division basis. Such a schedule could be established by initially assuming that all time slots at all transceivers are available for data traffic. Then, as each voice call is admitted, it would remove one transceiver from the pool available for data transmission at each node along the multihop path; when calls are completed, the transceivers would become available for data again.

5.3 Performance Measures for Data

As in Section 5.1, we evaluate the effect of administering a voice admission-control policy. Now, however, we can evaluate the effects of the control policy on performance metrics that incorporate aspects of both voice and data.

5.3.1 Residual capacity

In [17] we introduced data metrics that are based solely on knowledge of the voice statistics. From the steady-state distribution of the voice-call process (obtained using the product-form solution) we can calculate the average number of active calls of type j, denoted $\bar{x}_j$. At any individual node i the expected "residual capacity" C_i is defined to be the expected number of transceivers available for data, i.e., not being used for voice traffic. Thus

$$C_i = T_i - \sum_{j=1}^{J} c_{ji}\bar{x}_j.$$

This definition is easily extended to a network-oriented performance metric. Since data traffic is free to use a resource (transceiver) whenever it is not occupied by voice traffic, the residual capacity of an integrated network with given voice-utilization rates is an indicator of potential network performance with respect to data, and is, hence, of primary importance in evaluating delay. We have considered two different measures of residual network capacity. The *mean data capacity*, denoted by C_{mean}, is the mean of the nodal residual capacity, where the average is taken over all nodes in the network. The *bottleneck data capacity*, denoted by C_{bot}, is the minimum (over all nodes) residual capacity in the network.

When data-traffic levels are not specified a priori, the residual data capacity is a reasonable measure of the impact of data traffic on overall system performance. This metric provides an indication of how much data traffic can be supported by a network, given its resources (i.e., number of transceivers at each node) and the voice traffic that is to be supported. However, this indication is very imprecise because, under the receivers-verified model, a node will be able to use all of its residual capacity for data only if a sufficient number of transceivers are available at its neighbors. In a radio network, a transceiver (which may be used for either transmission or reception but not both simultaneously) must be available at both the transmitting and receiving nodes. Thus the "usable residual capacity" would typically be lower than

the residual data capacity computed by considering the nodes individually as we do in the receivers-assumed model. To evaluate the usable residual capacity accurately in our receivers-verified simulation model, it would be necessary to keep track of the number of transceivers in use at all network nodes; thus a packet can be transmitted only when a receiver is available at the destination node.

Of the two performance measures, C_{mean} may be of more importance if the primary concern with data traffic is the average data throughput that is supported by the network. However, C_{bot} may be more appropriate if delay is of primary importance, since the expected delay at a node will be unbounded if the offered data rate at that node is greater than the residual data capacity available at that node.

5.3.2 An Approximate Delay Model

When data-traffic levels are specified, appropriate data performance metrics include the throughput actually supported by the network as well as the average delay experienced by data packets. To facilitate the development of an approximation for delay, we have used the receivers-assumed model in which only the number of transceivers available at each transmitting node is of concern in determining the availability of data links, since it is assumed that a receiver is always available.

One possible approach is to model the data process at node i as an M/D/C_i queue. A simple expression for delay evaluation for the M/D/c queue is not available (a numerical procedure to determine its probability mass function and waiting time is described in [7]), and, furthermore, C_i will in reality be time-varying, in which case analysis becomes prohibitive. Thus we choose to oversimplify the model and assume that its performance can be approximated roughly by that of an M/D/1 queue with service rate equal to c times that of the M/D/1 system. We recognize that this approximation is poor; however, as we demonstrate in Section 10.2, even this poor approximation is extremely effective in ranking the performance of a number of policies. It is well known that the expected delay of an M/D/1 queueing system (not including service time) is

$$D = \frac{\rho}{2\mu(1-\rho)},$$

where $\rho = \lambda/\mu$ is the average load, μ is the service rate, and λ is the Poisson arrival rate. Thus we have used the following expression to approximate the data-packet delay at node i

$$\tilde{D}_i(\Omega) = \frac{\rho_i^d}{2C_i\left(1-\rho_i^d\right)},$$

where $\rho_i^d = \sum_{k\in i}\lambda_k^d / C_i$, and μ is set equal to 1 without loss of generality (i.e., implicitly, we assume that delay is measured in slots, where a slot is the time required by a transmitter to transmit one fixed-length data packet (all transceivers transmit at the same rate)). Here, $\sum_{k\in i}\lambda_k^d$ is the total data arrival rate at node i, where the sum is taken over all queues k that reside at node i. In the subsequent discussions, the delay metric we have used is the average nodal delay taken over all nodes.

This delay model is deficient in several ways. In addition to the fact that it ignores the need to pair transceivers at the transmitting and receiving nodes, it also ignores other critical aspects of network operation. Most significant is the implicit assumption that the number of data packets that are serviced per unit time at node i is constant at C_i (the fact that C_i is not, in general, integer-valued is a minor point here), whereas the number of servers available in the real system varies, based on the voice state. An accurate estimate of delay would have to take into account not only the expected voice state (which determines residual capacity) as we do in our model, but also the fraction of time spent in each voice state

as well as the statistics of the duration of each state visit.[12] Thus, the delay estimate based on this model is not accurate. However, as stated above, we show in Section 10.2 that this estimate does provide a remarkably accurate indication of the relative performance of a large number of different policies (and thus of their rankings), as demonstrated by SC simulations.

6. SC SIMULATION OF CIRCUIT-SWITCHED VOICE-ONLY NETWORKS

We have used SC techniques to evaluate the performance of the circuit-switched network of Fig. 8 under a number of different admission-control policies. The availability of the exact solutions obtained in [16, 17] for the present case of voice-only operation has allowed us to validate the results of our simulations. The discrete parameters of interest in this case are the circuit thresholds X_j and the linear-combination constraints Y_l. The events that must be generated are arrivals and departures. An arrival to circuit j is denoted by a_j, and occurs at a rate of λ_j^V. The departure rate for each active call on circuit j is μ_j^V; thus, if there are x_j active calls on circuit j, the departure rate for calls of type j is $x_j \mu_j^V$. This situation is translated to events in the simulation as follows. We define X_{jmax} to be the maximum value X_j can have over the entire set of policies. Thus

$$X_{jmax} = \min_{i \ni c_{ji}=1} (T_i).$$

Because there may be up to X_{jmax} active calls on circuit j, we must consider X_{jmax} different departure events d_{jn} ($n = 1, \ldots, X_{jmax}$) for circuit j, namely

$$d_{jn} \Rightarrow \text{feasible departure from circuit } j \text{ if } x_j \geq n, \text{ otherwise fictitious event.}$$

Following the usual technique used in uniformization, the *maximal rate* of this system is

$$\Lambda = \sum_{j=1}^{J} \left(\lambda_j + X_{jmax} \mu_j \right).$$

The upper part of Fig. 9 shows the ratio yardstick for the network of Fig. 8, where the maximum threshold on each circuit is three. This ratio yardstick corresponds to operation of the network at the maximal rate, and it can generate all events that are needed for the simulation of systems with control policies in which the circuit thresholds do not exceed 3. In our example $\lambda_j^V = \mu_j^V = 1$, which implies that all events are equally likely. All arrival events in this system are real, and the corresponding calls are accepted as long as their acceptance does not violate any of the system constraints or linear-combination constraints. However, departure events are fictitious if an insufficient number of calls of that type are currently active, as discussed above. For example, referring again to the upper part of Fig. 9, an event of type d_{13} will be fictitious (and hence ignored, although time will be updated) if there are less than three calls of type 1 currently active.

To reduce the number of fictitious events (and thus further improve the efficiency of the simulations) we have considered the use of "custom" ratio yardsticks, which are constructed to conform with the particular parameters of each sample path. The lower part of Fig. 9 shows a custom ratio yardstick for an example in which the circuit thresholds have been lowered to {1,3,2,3,1}. A number of events have been eliminated because they will always be fictitious, thus resulting in a reduction in the

[12] The fraction of time spent in each state is easily determined from the product-form solution. However, the expected duration of each state visit is harder to estimate because it is not reflected in the product-form characterization.

event rate from $\Lambda = 20$ to $\Lambda' = 15$; e.g., with the threshold on circuit 1 reduced to 1, the departure events d_{12} and d_{13} will always be fictitious. Clearly, by eliminating them from the ratio yardstick we can increase the percentage of real events. The remaining events are again equally likely, although there are only 15 of them instead of 20.

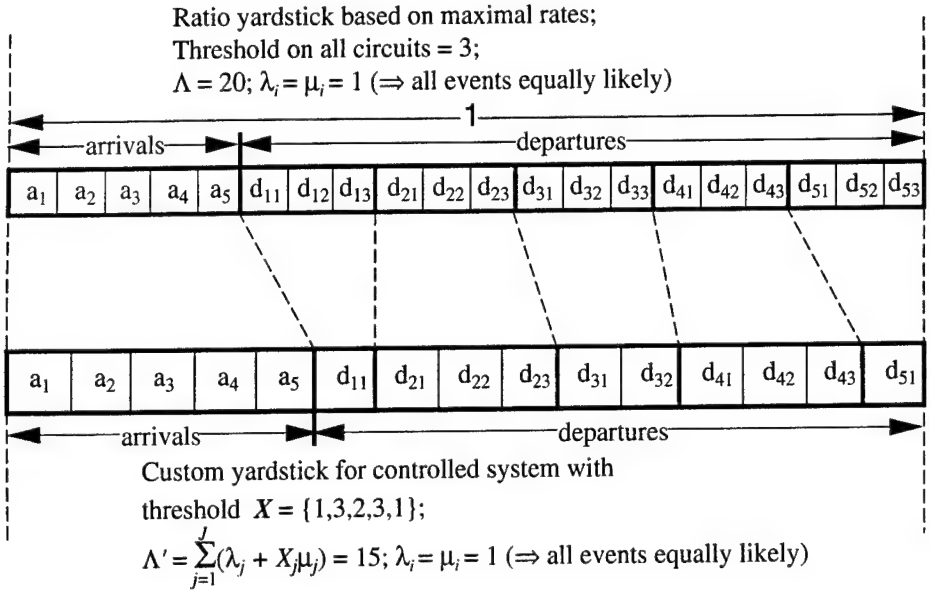


Fig. 9 – Ratio yardsticks showing event sets for the circuit-switched network

Despite the potential advantages of custom ratio yardsticks, we have found in our SC simulations that the added burden of maintaining and accessing separate custom yardsticks for every sample path outweighs the reduction in the number of fictitious events provided by their use [11]. However, there are some applications in which their use is beneficial, e.g., in cases where the percentage of events that are fictitious is high.

6.1 Simulation Timing Results

We have performed both BF and SC simulations of this network model to obtain timing results. In this section we briefly discuss the increase in simulation efficiency made possible by the use of the SC method, and in Section 9.1 we compare the network performance statistics (blocking probability) from our simulations with the exact values given in [16]. Performance results presented in [24] and [11] show that when more than 2,000 sample paths are simulated, the SC method is approximately 2.5 times faster than the BF approach.

Despite the 17.5-fold increase in the size of the event set for the circuit-switched voice model as compared to the M/M/1/K model (there are 35 events in the voice model as opposed to 2 in the M/M/1/K model), the time per call for the event-generation and state-updating functions is approximately the same as that in the M/M/1/K simulations. The speedup, on a per-event basis, is $S_e(100) = 6.08$, which is commensurate with the M/M/1/K result. However, in the voice network with $\rho^V = 1$ for each circuit, approximately 1/3 of the events generated in the SC simulation are fictitious. Because our BF simulation uses custom alias tables to reduce the number of fictitious events, the *predicted speedup*, which must take into consideration the number of events generated in the two approaches, is $S_e(100)\cdot(46821/68920) = 4.13$. As the problem is scaled up, the SC approach suffers further because it must allocate memory for all sample paths simultaneously (which, for large N, results in main memory paging and data cache breaks), while it suffices to allocate memory for a single sample path in the BF approach, regardless of the number of sample paths being simulated. Based on these results, we have found that the time per call

to the state-update and the event-generation functions is relatively insensitive to the size and complexity of the problem. Thus, the SC approach should scale reasonably well, provided that sufficient memory is available.

7. SC SIMULATION OF INTEGRATED VOICE/DATA NETWORKS

Now we demonstrate that it is straightforward to extend our simulation to the case of an integrated network such as that shown in Fig. 8a, in which packet-switched data with fixed-length packets is supported along with circuit-switched voice of exponential call duration. Although the SC approach is directly applicable only when the interevent times in the system are exponentially distributed, Ho et al. [9] have shown that certain deterministic events, which they refer to as "standard control clock (SC^2)" events, can also be incorporated into the simulation. In addition, Chen and Ho [10] have demonstrated the use of shift-exponential and hyper-exponential distributions to model nonexponential distributions. In this section we describe a different method for the incorporation of deterministic events (fixed-length data-packet transmission completions, henceforth termed "data-packet departures" or "deterministic events" interchangeably) into SC simulations, which was introduced in [11, 25].

The stochastic events consist of voice-call arrivals and departures, and data-packet arrivals. They are generated at a maximal event rate of

$$\Lambda = \sum_{j=1}^{J} (\lambda_j^V + X_{jmax}\mu_j^V) + \sum_k \lambda_k^d,$$

in the usual way, i.e., by drawing two random numbers (one to determine the interevent time and the other to determine the event type by means of the alias method). Here X_{jmax} is the maximum threshold on calls of type j as defined earlier. The system state $(\boldsymbol{x}, \boldsymbol{d})$ is updated whenever one of the stochastic events occurs. Since the deterministic data-packet departure events are not included in the stochastic event stream, they require special processing. First we discuss what appears to be a natural and direct way of processing deterministic events, and then present an alternative, more efficient, processing method.

7.1 A Direct Processing Method

Deterministic events will be naturally scheduled and processed when the state is updated as a result of the occurrence of a stochastic event. When a deterministic event is scheduled, it is time-stamped and placed in a deterministic-events queue. Hence, in each sample path, in addition to the data-packet queues, we establish a single separate queue for dealing with deterministic events. Then, before processing each stochastic event, all eligible events[13] in the deterministic-events queue will be processed. More precisely, deterministic-event scheduling occurs when a data-packet transmission begins, i.e., when

- A data packet arrives at a link with available transceivers (i.e., a transceiver is available at both the queue's source and destination nodes); or
- A data-packet departure provides a link with available transceivers for queued data (provided that the channel is not seized by a voice call); or
- A voice departure provides all the links along the circuit path of that voice call with available transceivers for queued data.

[13] An eligible event is an unprocessed deterministic event that is scheduled to occur before the next stochastic event.

An "available" transceiver is one that is not committed (at that time instant) to supporting either voice or data traffic, and hence is available to support the transmission of the packet in question.[14] Unlike the simplified analytical model of Section 5.3, the simulation model can check for the availability of the necessary transceivers at the receiving nodes. Since time is continuous in the simulation, there is no possibility of two events happening simultaneously, and thus no ambiguity as to which data packet would be transmitted. As noted earlier, the acceptance of a voice call depends only on the voice-traffic state. Voice calls cannot be blocked by data packets that are using transceivers needed for the call; we simply assume that the data packets complete their transmission, after which the voice call claims all needed resources.

7.2 An Improved Processing Method

Instead of implementing the deterministic-events handler as just described, i.e., by maintaining the additional deterministic-events queue, we can significantly improve the efficiency of the implementation by taking advantage of the network traffic characteristics, as follows. First we observe that, because the voice state is constant between consecutive voice (arrival or departure) events, the data service rate (i.e., the residual capacity available for data) is also constant for that time interval. Therefore, immediately before processing the next voice event, it is possible to reconstruct the sequence of deterministic events that occurred since the last voice event. This means that, instead of checking the deterministic-events queue and processing eligible events before *every* stochastic event, we need to process deterministic events only before all voice events. We do not need to process deterministic events before data-packet arrival events, because data arrivals do not change the data service rate. Arriving data packets are simply stored in the appropriate queue. This method potentially yields significant time savings, because the data arrival rates can be orders of magnitude greater than the voice arrival rates, in which case the number of data arrival events (each of which corresponds to a savings of one deterministic-events processing session) is orders of magnitude greater than the number of voice events.

The ability to reconstruct the data service process from the voice-state process and the queue sizes eliminates the need to maintain a deterministic-events queue and to perform the time-stamping functions mentioned earlier. With the efficient implementation, deterministic-events processing for each queue simply involves decreasing the size of that queue by either the product of that queue's link capacity (which is the minimum of the number of available transceivers at the link's source or destination node) and the number of slots since the last voice event (one slot = time required to transmit one data packet), or the number of packets in the queue, whichever is less. There is also some additional bookkeeping involved to prevent two data queues from simultaneously using the same resources. This method eliminates the need to time-stamp, because we update the queue size statistics with each voice event; since the voice-events are generated by a Poisson process, we use the PASTA (Poisson Arrivals See Time Averages) theorem to measure the average queue size, and Little's result to obtain the expected delay. It follows that with this method the queues are observed less frequently, thus suggesting that it may be necessary to run the simulation longer to achieve the accuracy delivered by the less-efficient direct processing method described earlier. However, our performance results in Section 10.2.2 demonstrate that the quality of the policy rankings obtained is virtually indistinguishable from that obtained by using the direct-processing method. Furthermore, the increase in simulation efficiency allows the latter implementation to perform much longer runs (in terms of number of events) and thereby to achieve a high degree of accuracy in less time than is required by the former method.

---

[14] As noted in Section 5.2, our model does not address "protocol" issues such as the transmission of control packets to coordinate the transmitting and receiving nodes. We assume an idealized model in which the nodes pair themselves as needed, without incurring any time delay or overhead.

7.3 Integrated Network Simulation Timing Results

We have measured the time required to perform both BF and SC simulations of N different admission-control policies in the integrated voice/data network. In these simulations, we have observed some interesting and previously undocumented behavior. Initially we used an HP Apollo Series 700 Model 750 workstation to perform these 10^5-event simulations. When events are passed one at a time to the sample paths (the regular SC method described earlier), the maximum value of speedup that has been observed for this problem is 4.5, which is obtained for $N = 50$, as shown by curve (a) in Fig. 10; a precipitous drop in speedup is observed as N is raised from 64 to 68, followed by a gradual reduction until the speedup hovers around 2.3 for values of N between 3,000 and 12,000. When N becomes excessively large, i.e., $N \geq 12{,}250$, the simulation crashes as memory constraints are exceeded. All of the reduction in speedup is attributable to a slowdown of the SC simulation run; the computation time required by the BF method is unaffected and remains directly proportional to the number of sample paths. This is shown in Fig. 11, where the curve of BF simulation time has a virtually constant slope, but the SC simulation curve has a pair of discontinuities near $N = 66$ (which correspond to the precipitous drop in speedup as N is raised from 64 to 68) and an increase in slope for $N > 68$.

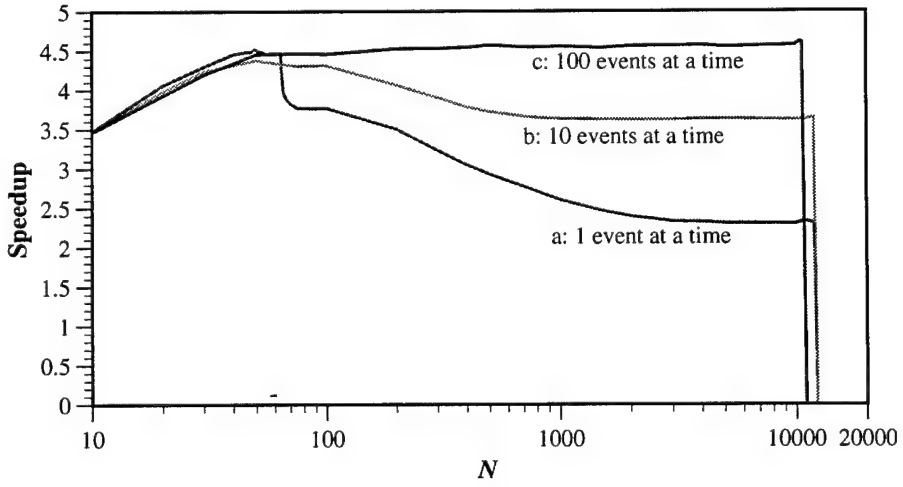


Fig. 10 – Speedup using an HP workstation in simulations of N admission-control policies in the integrated network

These results are repeatable and do not depend on which policies are examined, but only on the *number* of policies (and hence sample paths) that are examined simultaneously. The originally unanticipated decrease in speedup for N greater than 50 is consistent with the occurrence of cache breaks, i.e., once the memory requirement for N sample paths exceeds the data cache size, the CPU must transfer data representing one sample path from main memory to the data cache N times per event; thus there are N cache breaks per event. The subsequent gradual reduction, as N is increased further, may be the result of memory paging. For very large N, the computer may thrash with input/output operations dominating the timing considerations. In contrast, the memory requirements of BF simulation are independent of N since only one sample path at a time is simulated. Based on these observations, in our example and on this platform it would be best to apply the SC method to blocks of 50 sample paths. In general, some experimentation would be needed to determine the optimal number of sample paths to simulate simultaneously; this number depends on the particular problem being investigated as well as on the characteristics of the computing platform and the particular way in which the simulation is programmed. On any other machine a different critical threshold value of N can be found beyond which the efficiency of SC decreases. Furthermore, for excessively large values of N the memory requirements of SC simulations will exceed the available resources, and the SC simulation will either crash or thrash. From

these studies, we have concluded that the determination of the optimal number of sample paths is an important issue in SC simulation on sequential machines, one which is beyond the scope of this report.

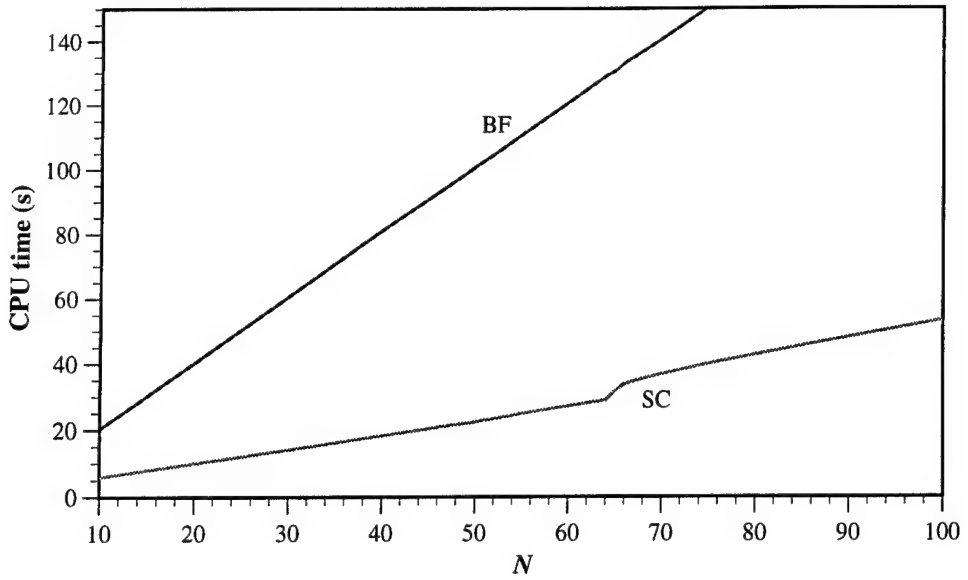


Fig. 11 – CPU time required to perform 100,000-event BF and SC (1 event at a time) simulations of N sample paths on an HP workstation

To bypass the deleterious effects of the cache breaks, we have considered a modified form of SC simulation in which groups of events, rather than individual events, are passed to each sample path. Curve (c) in Fig. 10 shows that when groups of 100 events are passed to each sample path, there is little dependence of speedup on N once it has reached approximately 50 (until the simulation crashes at N = 11,000, which is 1,250 fewer sample paths than the maximum N for the case in which only one event at a time is passed). The effect of cache breaks and memory paging is reduced significantly when it is amortized over the 100 events that are processed together. Cache breaks are not eliminated, but 100 times as much work is accomplished with each cache break. Hence, grouping events together mitigates the platform-induced limitation of speedup. We expect that the speedup is relatively insensitive to the event group size, as long as it is large enough so that the impact of cache breaks per event is sufficiently small. As shown by curve (b) in Fig. 10, less speedup is achieved when a smaller group size is used. However, excessively large event group sizes result in excessive memory requirements to store the event sequence. The cache break problem is encountered again, but in this case it is a result of cycling through the event sequence rather than cycling through the set of sample paths.

The results presented in Figs. 10 and 11 are based on the use of an HP Apollo Series 700 Model 750 workstation (denoted "HP"). As shown in Figs. 12 and 13, qualitatively similar results to those obtained on an HP have been obtained on a Sun SPARCstation 10 (denoted by "Sun" in the figures), and on a Silicon Graphics Indigo2 XZ workstation (denoted by "SGI" in the figures), although the speedup and threshold value of N are somewhat different. Table 4 shows that the Sun computer has the smallest memory and data cache of the three platforms examined, and thus it can benefit most from speedup improvements obtained by passing groups of events to each sample path. Although Fig. 12 shows that of the three platforms examined, the Sun achieves the greatest speedup when passing groups of 100 events at a time, Fig. 13 shows that the Sun is actually the slowest machine. The maximum number of sample paths that can be simulated simultaneously when using the SC approach on this machine is about 4000. As can be seen in Fig. 13 (a), the time required to perform SC simulations of 5000 sample paths is more than twice that required for 4000. To explain this behavior, we again address the impact of memory considerations. When the simulation memory requirements exceed some fraction of the system RAM,

blocks of the simulation process (or the entire simulation process) are swapped to disk to give the system and other concurrent processes the memory needed to use their CPU cycles. Higher memory requirements require more frequent swapping, so that, e.g., when performing SC simulations of 5000 sample paths, the CPU usage is dominated by performing swaps and the simulation slows to a crawl. This phenomenon is known as "thrashing." In Fig. 12, the speedup value for "1 event at a time" SC simulation is *estimated* to be very near zero for $N = 4000$ because the simulation was interrupted before its normal completion; the SC simulation did not crash, but its thrashing prevented any significant progress in an overnight run. Hence, we presume that the simulation would have eventually achieved normal completion, and yielded a speedup value near zero.

Table 4 – Memory on Three Platforms

	HP	SGI	Sun
RAM (MBytes)	128	96	32
Data Cache (KBytes)	256	8	20
Instruction Cache (KBytes)	256	8	16
Secondary Cache (KBytes)	0	1000	0

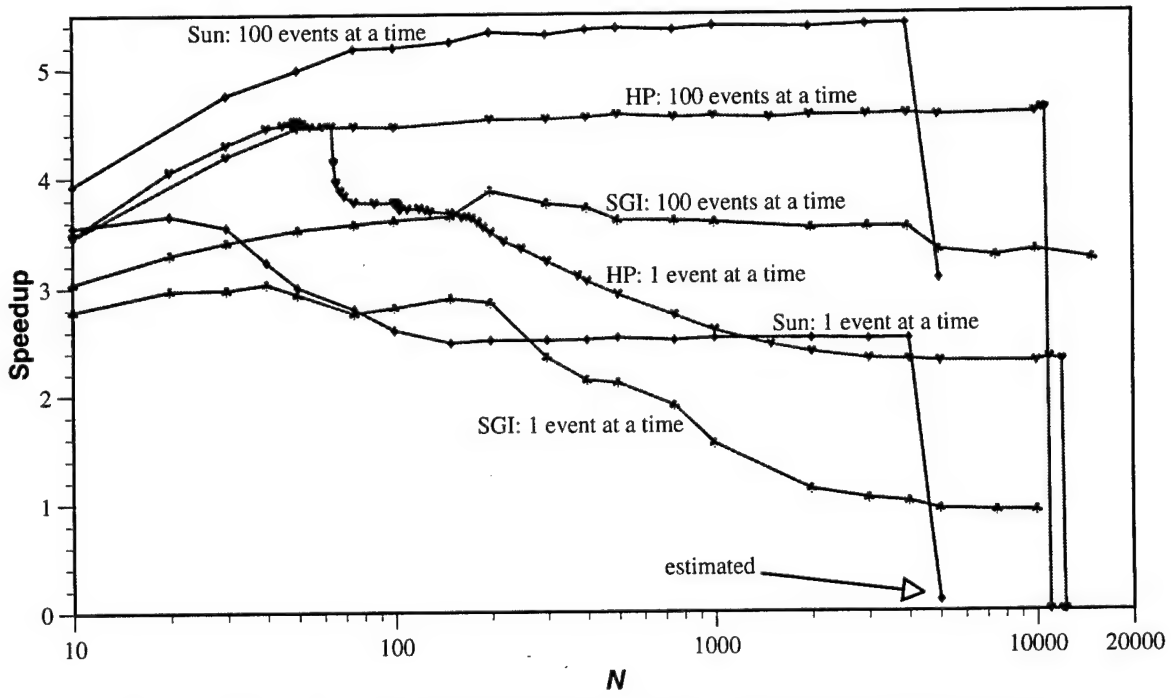
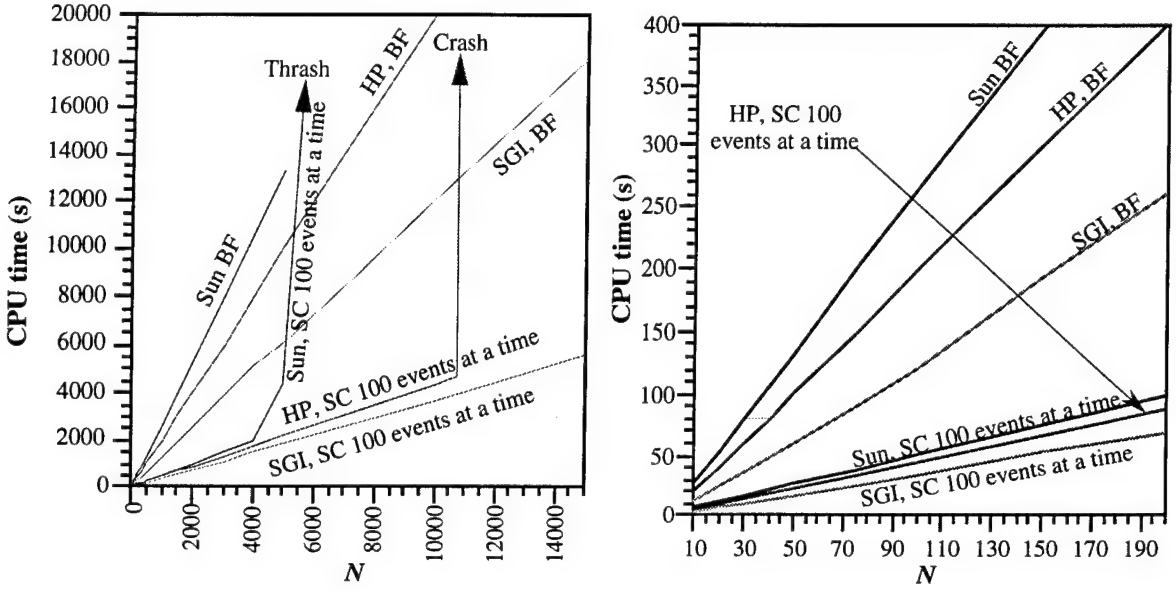


Fig. 12 – Speedup on Sun, HP, and SGI platforms obtained by using SC simulation of N sample paths (1 event at a time, and 100 events at a time)

Because the HP computer has more RAM and data cache than the Sun (Table 4), when performing SC simulation and passing one event at a time the HP achieves its peak speedup at a higher value of N than does the Sun, as shown in Fig. 12. Both the "1 event at a time" and the "100 events at a time" simulations on the HP achieve a fairly constant speedup for $N > 2000$. This speedup is maintained with increasing N until the machine's memory capacity is exceeded and the simulation crashes. The SGI has nearly as much RAM as the HP, but it has a two-layer cache system with 8 Kbytes of primary data cache plus 1 MByte of secondary unified instruction/data cache. We conjecture that this layered cache scheme is responsible for the two distinct roll-offs in speedup shown in Fig. 12 for the "1 event at a time" simulation. When this speedup curve falls below a value of 1.0, the BF approach is actually faster than the SC simulation. The performance on all three platforms degrades once N exceeds some platform-

dependent value. However, the SGI's speedup performance degrades more gracefully than the others. Rather than crashing (HP) or thrashing (Sun), the SGI suffers a gradual decrease in speedup with large, increasing N values (although there is almost surely some value beyond which the SGI will crash or thrash too). Figure 12 shows that two benefits are achieved by passing groups of 100 events at a time to each sample path on the Sun and the SGI. First, the reduction in speedup that occurs as N increases is mitigated. (This benefit is seen on the HP as well.) Second, the speedup achieved for small values of N is increased.



(a) the big picture (b) expanded view for small N

Fig. 13 – CPU time required to perform BF and SC simulations of N sample paths

Our timing results for simulations of the integrated network suggest that the SC method scales well with increasing problem complexity. The integrated network is considerably more complex than the voice-only network. In addition to the 35 voice-event types, there are ten types of data-arrival events and their corresponding deterministic data-departure events (with data events occurring at a considerably higher rate than voice events). Despite this added complexity, the time required to simulate the integrated network is only slightly larger than that for the voice-only network. This further confirms our observations on scalability made in Section 6.1 when comparing SC simulations of the voice-only network to those for the M/M/1/K queue.

8. ORDINAL OPTIMIZATION

Ordinal optimization [4] is the determination of control policies that perform relatively well compared to other candidate policies without necessarily obtaining accurate estimates of the performance values associated with these policies. The motivation underlying ordinal optimization is that finding the optimal solution (or control policy) is often too costly or time consuming, although a suboptimal solution (that may be found quite easily) may provide sufficiently good performance. Also, it is not necessary to obtain accurate estimates of system performance values associated with these policies, if an accurate ranking of these policies can be determined.

In the literature, several different approaches have been considered for ordinal optimization. In the original paper on this subject [4], short simulation runs were used in conjunction with SC techniques to obtain an approximate ranking of control policies, and it was observed that many of the high-performance

policies also perform well over the long run. It was also observed that, when the search space is too large to be examined exhaustively, acceptable performance can be obtained by examining a randomly chosen subset of policies. The principles of order statistics [26] suggest that if a sufficient number of such randomly chosen policies are examined, we will encounter some good ones, which can then be examined in greater detail. One way to do so is to perform a longer simulation run for each of these.

Other approaches to ordinal optimization have also been proposed for different types of problems. For example, the simulation time in systems with rare events can be prohibitive if standard brute-force techniques are used. However, the use of a surrogate design problem, in which the events of interest are not so rare, can speed up the simulation considerably while providing good ordinal rankings of control policies [27, 28]. For example, the policy that optimizes system performance based on the minimization of the probability of buffer overflow for a relatively small buffer size (a not-so-rare event) may also minimize the probability of buffer overflow for larger buffer sizes (which may be rare events), a property referred to as "ordinal equivalence" [27]. The first use of ordinal optimization in conjunction with SC techniques on a single-instruction, multiple-data (SIMD) machine was demonstrated in [29], where it was shown that a reduction in simulation time of several orders of magnitude could be achieved.

In this report we implement ordinal optimization in three different ways. In the first we use relatively short simulation runs (as in [4]) to obtain a ranking of a number of policies in terms of voice-call blocking probability. We find that, although the measured performance may not be very accurate, the ranking of policies is relatively immune to the effects of "estimation noise." Our second approach, which we use for data-packet delay, is the use of crude analytical models that capture the crucial aspects of system behavior. Again, remarkably accurate policy rankings are achieved (in this case without using simulation at all), although the performance estimates are poor. Our third approach, again for data-packet delay, is the use of imprecise simulation models that, like the crude analytical models, incorporate the salient features of the communication model.

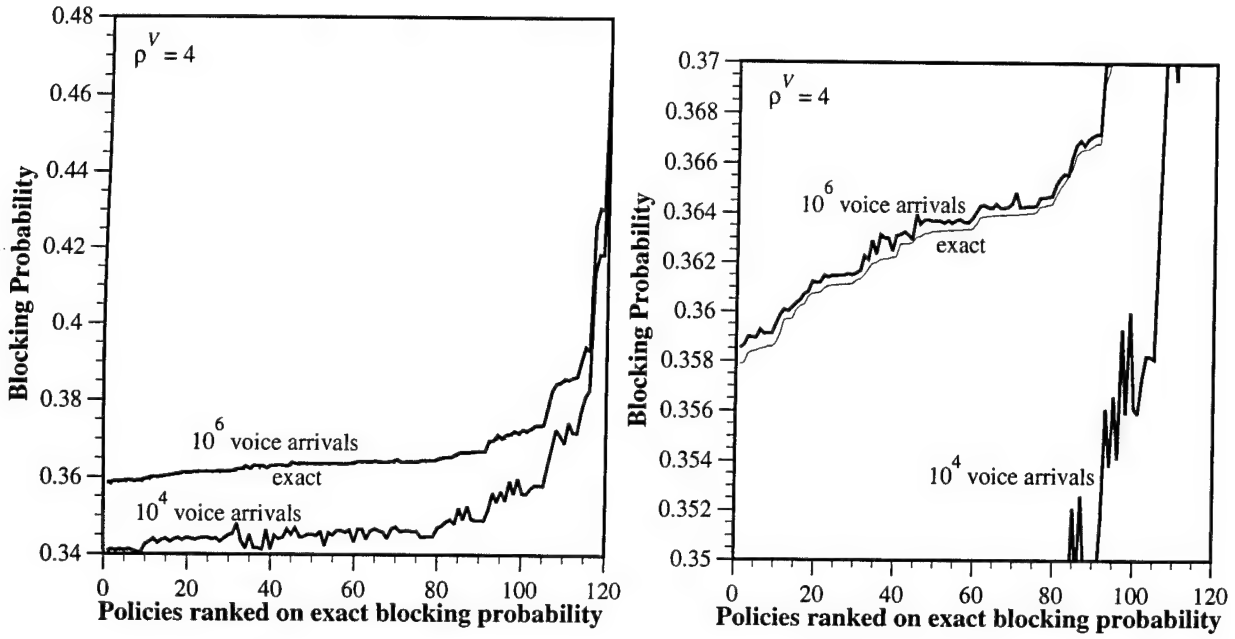
9. VOICE-CALL BLOCKING PROBABILITY: PERFORMANCE EVALUATION AND ORDINAL OPTIMIZATION

We have performed SC simulations of the integrated network shown in Fig. 8a by using the model discussed in Sections 5 and 7. Eight transceivers are assumed present at each node in the network. Recall from Section 5.1 that a policy for this network can be written as $\Omega = \{X_1, \ldots, X_5, Y_1, \ldots, Y_5\}$. We have simultaneously evaluated the 120 different control policies $\{X_1, 6, 6, 6, X_5, 8, 8, Y_3, 8, 8\}$, where $X_1, X_5 = 0,\ldots 6$; $Y_3 = 0,\ldots,8$; $Y_3 \leq X_1 + X_5$; $X_1 \leq Y_3$; and $X_5 \leq Y_3$. We have used various values of $\lambda_j^V = \lambda^V$, $\mu_j^V = \mu^V, j = 1, \ldots, 5$, all of which are subject to $\lambda^V/\mu^V = \rho^V = 4.0$. For data, we have set $\lambda_i^d = \lambda^d = 5$, $\mu_i^d = \mu^d = 10$, to yield $\rho_i^d = \rho^d = 0.5$, $i = 1, \ldots, 10$. To assess the effectiveness of our simulation techniques, we have evaluated the accuracy of the simulation-based values of the desired performance measures, as well as the accuracy of the corresponding ordinal rankings of the policies.

9.1 Evaluation of Voice-Call Blocking Probability

In Fig. 14(a) we show the exact and simulated voice-call blocking probability associated with the 120 control policies. The exact results are determined numerically from the product-form solution, and simulated results are based on runs of 10^4 and 10^6 voice-arrival events. Figure 14(b) is an expanded view of the same curves, which permits a more-detailed comparison of the results. As noted earlier, the results for blocking probability are independent of data-traffic parameters; in fact, they do not depend on the individual values of λ_j^V and μ_j^V, but only on the ratios $\rho_j^V = \lambda_j^V/\mu_j^V$. The horizontal axis is simply the ordering from the best (minimum blocking probability) policy to the worst, based on the exact model; thus blocking probability is a monotonically nondecreasing function of the horizontal axis. It is apparent

from the closeness of the curves in Figs. 14(a) and 14(b) that the results of the longer simulation are extremely accurate; the simulation error is never more than 0.2%. It is also apparent that the shorter simulation is not long enough to predict blocking probability accurately.



(a) the complete picture (b) an expanded view

Fig. 14 – Probability of blocking across the range of policies

9.2 Ordinal Ranking of Policies in Terms of Voice-Call Blocking Probability

In Fig. 15 we compare ordinal rankings of the voice blocking probability obtained from two SC simulations to the exact rankings. For these studies we used $\lambda^V = 4$, and $\mu^V = 1$. The two SC simulations differed only in their durations, which were based on 10^4 and 10^6 voice-call arrivals. The ordinal ranking of the simulation results shows remarkable agreement with the exact ordinal ranking (ideally the curve would be a straight line with unit slope). For long simulations this is not surprising; however, it is remarkable that the short simulation placed eight of the top ten policies in the top ten positions. This agreement is achieved despite the insensitivity of blocking probability to the policy used. It is apparent from Fig. 14 that there is little sensitivity to the policy that is used when blocking probability is the only performance measure of interest. For example, as the policies are examined from the best to the 80th out of 120, the blocking probability increases by only a small amount, i.e., from 0.358 to 0.365.

9.2.1 A Performance Measure for Ordinal Rankings: The Spearman Rank Correlation Coefficient

As Fig. 15 shows, the policy rankings are considerably better for the longer simulation. To facilitate the comparison of sets of rankings, it is helpful to use a quantitative measure of the quality of a set of rankings. To provide the basis for such a metric, let Ξ_i be the performance measure value (in this case blocking probability) associated with policy i ($1 \le i \le N$), based on a simulation run; and let Ψ_i be the exact performance measure value associated with policy i (where the index i is arbitrary, i.e., not based on performance measure values). Thus the pair (Ξ_i, Ψ_i) represents the simulated and exact performance values associated with policy i. Now define R_i = rank Ξ_i (where $R_i = 1$ means that policy i is the policy with the best value of Ξ_i) and S_i = rank Ψ_i. One method of comparing these sets of rankings is the use of Spearman's rank correlation coefficient [30]

$$r_s = 1 - \frac{6\sum_i (R_i - S_i)}{N(N^2 - 1)},$$

which provides a scalar measure of the "association" between two sets of rankings. For the case in which all of the simulated rankings are correct (i.e., $R_i = S_i$, $1 \le i \le N$), $r_s = 1$.

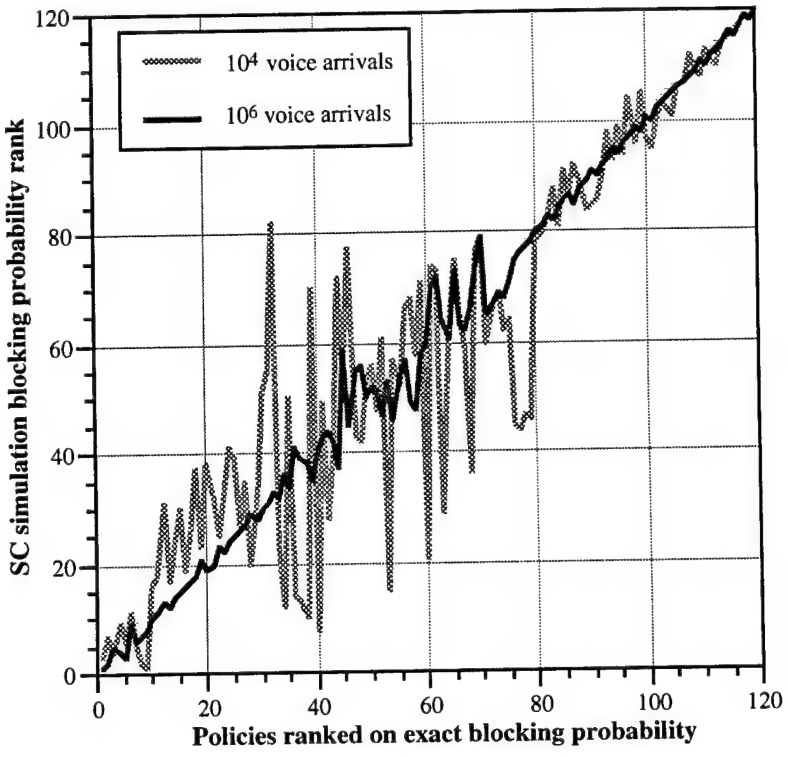


Fig. 15 – Blocking probability found in SC simulations compared to the exact value, $\rho^V = 4$

Figure 16 shows the improvement in ordinal ranking of blocking probability, based on r_s, as the simulation length is increased from 10^3 to 10^6 events. This figure shows that r_s tends to increase (although not monotonically) with increasing simulation length, eventually reaching a value close to the ideal of 1.0, thus implying that the set of rankings tends to improve as the duration of the simulation increases. Future studies will address the development of alternative metrics for the evaluation of performance rankings. For example, it may be appropriate to weight more heavily the best policies. Alternatively, instead of comparing the rankings of all policies, it may be more appropriate to compare the performance of a set of the best policies to the actual optimal performance value.

9.2.2 SC Ordinal Rankings Based on Several Random Seeds

To determine whether the quality of the ordinal rankings shown in Fig. 15 is typical of that obtained for short SC simulation runs, we performed SC simulation runs of different lengths, based on the use of six different initial random number generator seeds. Figure 17 shows blocking probability and ordinal rankings for runs of length 10^4; these figures represent the same simulation scenario as Figs. 14(a) and 15, except that different seeds are used. Similarly, Figs. 18 and 19 show blocking probability and ordinal rankings for simulations of 10^5 and 10^6 events, based on the same set of six random seeds. There is certainly a high degree of variation in the quality of performance estimates based on the random seed. However, for all random seeds a significant number of truly good solutions are among the best solutions

found in the simulations, even for the simulations of 10^4 events duration. As the length of the simulation grows, the accuracy of both the performance estimates and the ordinal rankings improves.

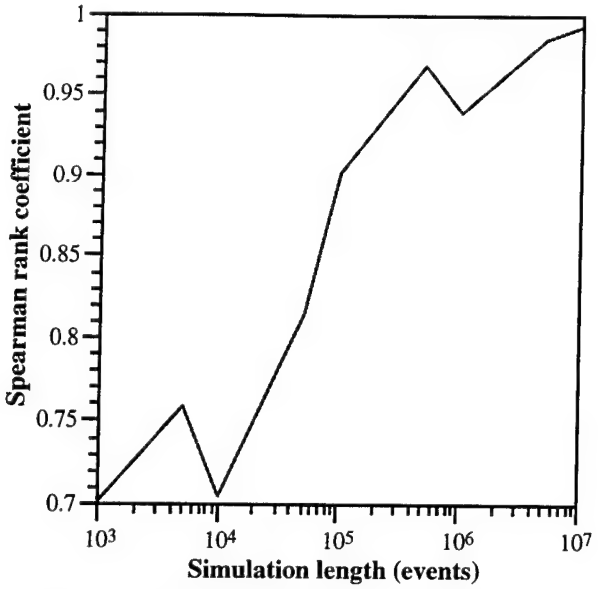
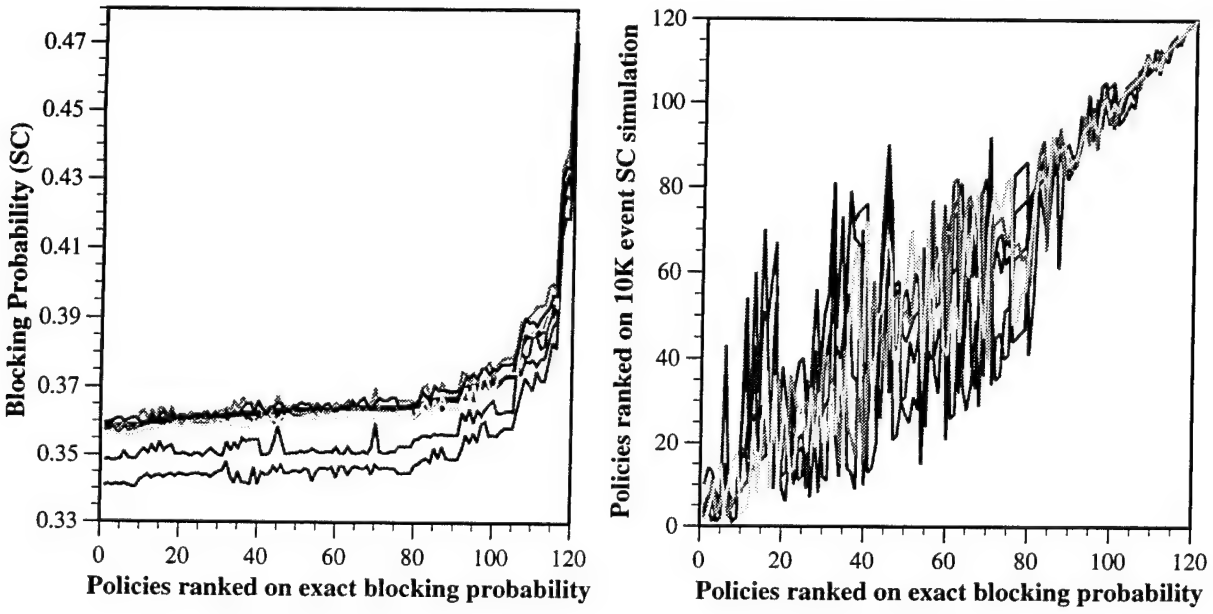


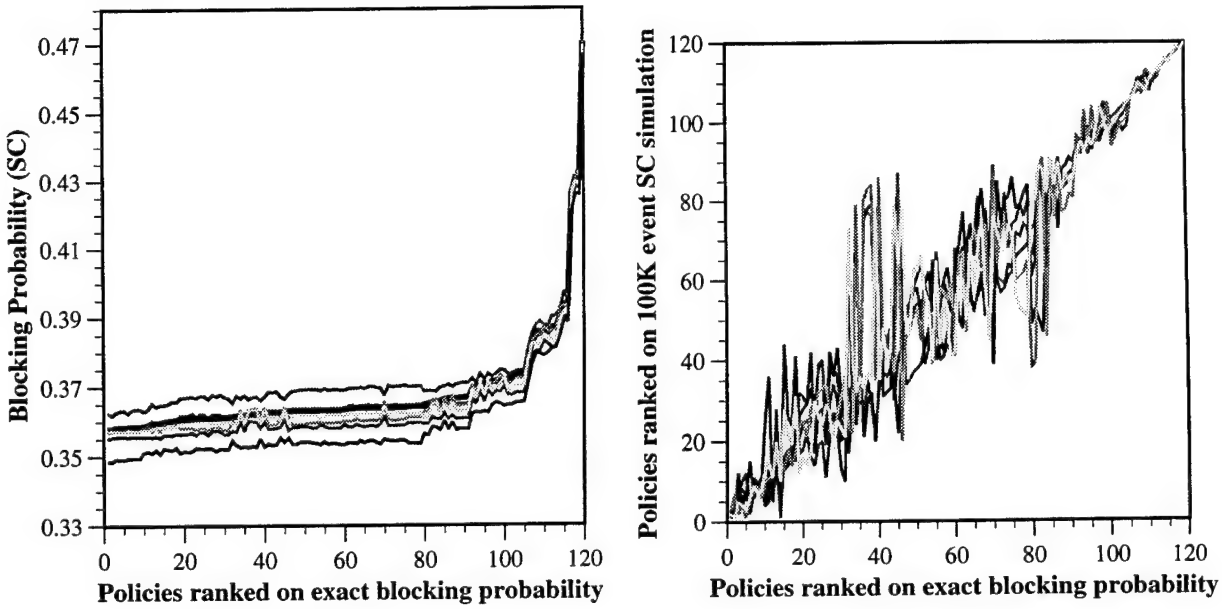
Fig. 16 – Quality of ordinal rankings of blocking probability, based on Spearman's rank coefficient for SC simulations of various lengths



(a) blocking probability

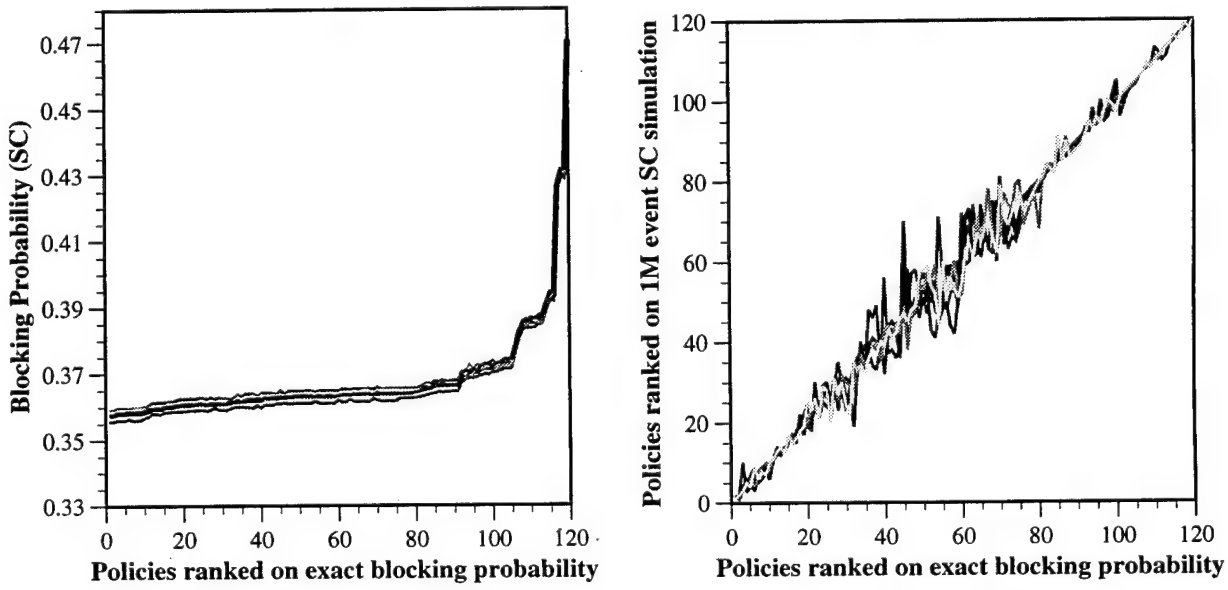
(b) ordinal rankings

Fig. 17 – Blocking probability and ordinal rankings for SC simulations; six random-number generator seeds; simulation duration = 10^4 events



(a) blocking probability (b) ordinal rankings

Fig. 18 – Blocking probability and ordinal rankings for SC simulations; six random-number generator seeds; simulation duration = 10^5 events



(a) blocking probability (b) ordinal rankings

Fig. 19 – Blocking probability and ordinal rankings for SC simulations; six random-number generator seeds; simulation duration = 10^6 events

9.2.3 The Impact of Common Event Sequences on Ordinal Rankings

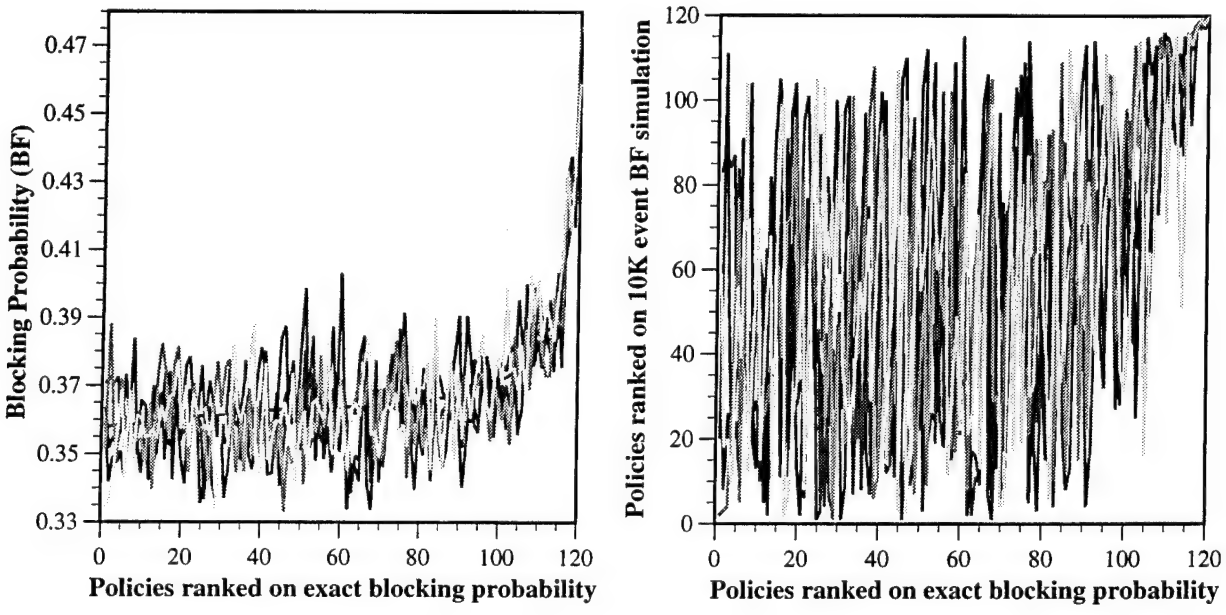
The SC method uses a common event sequence for all control policies; thus it permits the direct comparison of the performance of these policies under the same input conditions.[15] The benefits of the use of common event sequences for comparing policies has been noted in [8, 31-33]. In this subsection

[15] Alternatively, a common event sequence may be implemented for each policy by means of separate BF simulations in which the same random-number seed is used to initialize each simulation.

we demonstrate that the use of a common event sequence provides considerably better ordinal rankings than the use of independent sequences.

By examining Figs. 17 through 19 we can see that the use of different random-number generator seeds can produce differences in performance that are considerably greater than those resulting from the use of different admission-control policies, especially for short simulation runs (measured in terms of blocking probability, which is fairly insensitive to the control policy through the best 87 policies). To determine how the use of different event sequences impacts on ordinal rankings, we have evaluated blocking probability by means of multiple BF simulations in which independent event sequences are used to generate the sample paths for each admission-control policy. Figures 20 through 22 show blocking probability and ordinal rankings for six sets of random-number seeds, which correspond to the cases shown in Figs. 17 through 19. Note that the ordinal rankings are poor for a simulation duration of 10^4 events. For a simulation of length 10^5 events, some evidence of improved policy ranking is visible, although results are generally not acceptable. Finally, a trend toward reasonably accurate rankings is observed for the simulations of duration 10^6 events. For simulations of this length, performance is nearly independent of the random-number seed; this is evident from the closeness of the curves in Fig. 22(a). However, the quality of these ordinal rankings appears to be comparable to that for the SC simulations of duration 10^4.

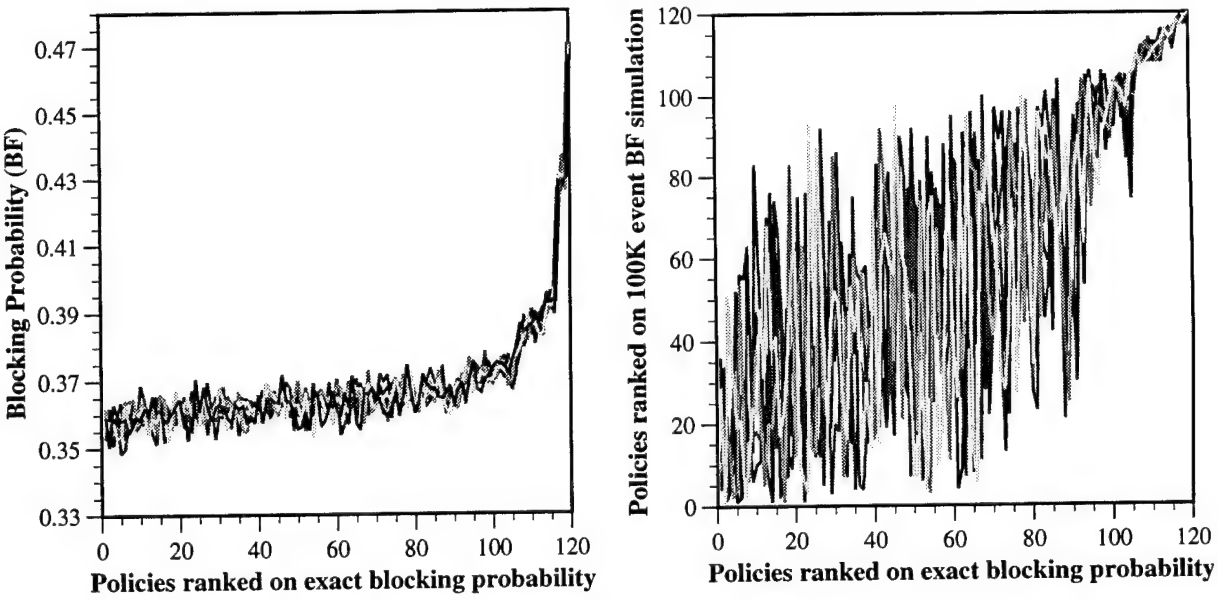
The failure of the independent simulations to provide good rankings for short simulation runs clearly demonstrates the benefits obtained by using a common event sequence. In our example, the use of independent event streams appears to require a simulation duration of about two orders of magnitude greater than that required for SC simulation. Thus, in addition to the benefit of simulation speedup (in terms of the time required to simulate a specified number of events), the SC technique's use of a common event sequence introduces correlation into experiments so that accurate rankings are obtained early in the simulation. Therefore, highly accurate ordinal rankings can be obtained with much shorter simulations than would be required with independent simulation runs.



(a) blocking probability

(b) ordinal rankings

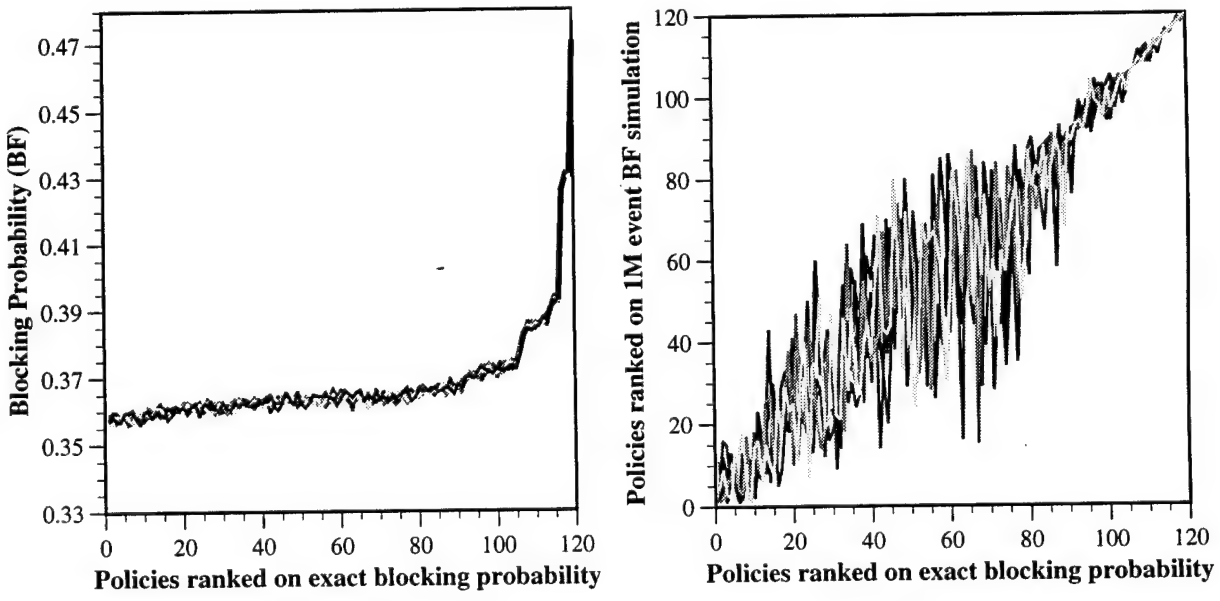
Fig. 20 – Blocking probability and ordinal rankings for independent BF simulations; six random-number generator seeds; simulation duration = 10^4 events



(a) blocking probability

(b) ordinal rankings

Fig. 21 – Blocking probability and ordinal rankings for independent BF simulations; six random-number generator seeds; simulation duration = 10^5 events



(a) blocking probability

(b) ordinal rankings

Fig. 22 – Blocking probability and ordinal rankings for independent BF simulations; six random-number generator seeds; simulation duration = 10^6 events

The Spearman rank coefficient for the examples shown in Figs. 17 through 22 is shown in Fig. 23. Based on this criterion, the SC results reflect a high degree of "association," even for relatively short simulation runs, whereas the independent BF simulation runs perform considerably worse based on this measure. These results are consistent with the observation made above that ordinal rankings based on independent simulations of duration 10^6 events are comparable to those of SC simulations of duration 10^4 events.

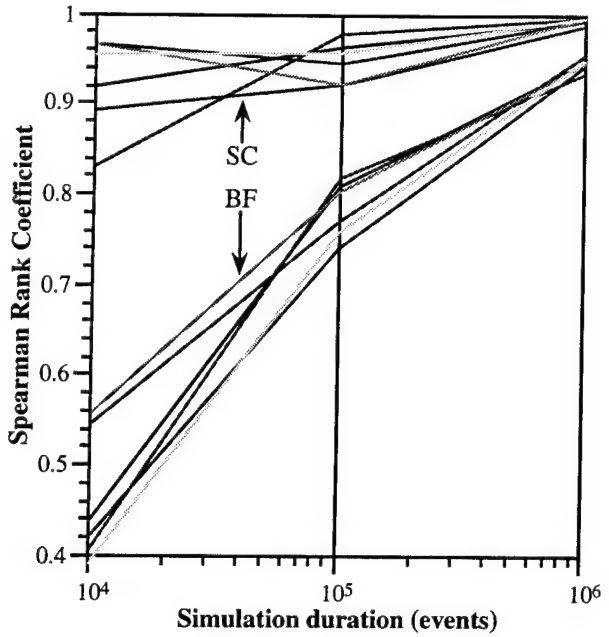
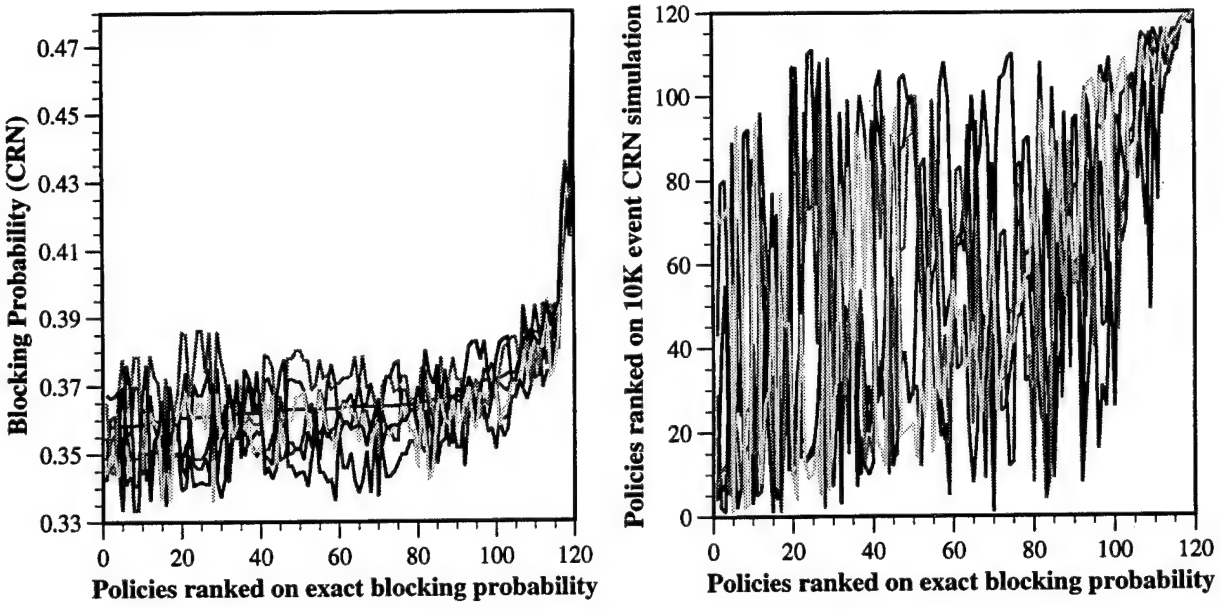


Fig. 23 – Quality of ordinal rankings of blocking probability, based on Spearman's rank coefficient for six SC simulations and six independent BF simulations of various lengths

9.2.4 The Use of Common Random Numbers But Different Event Sequences

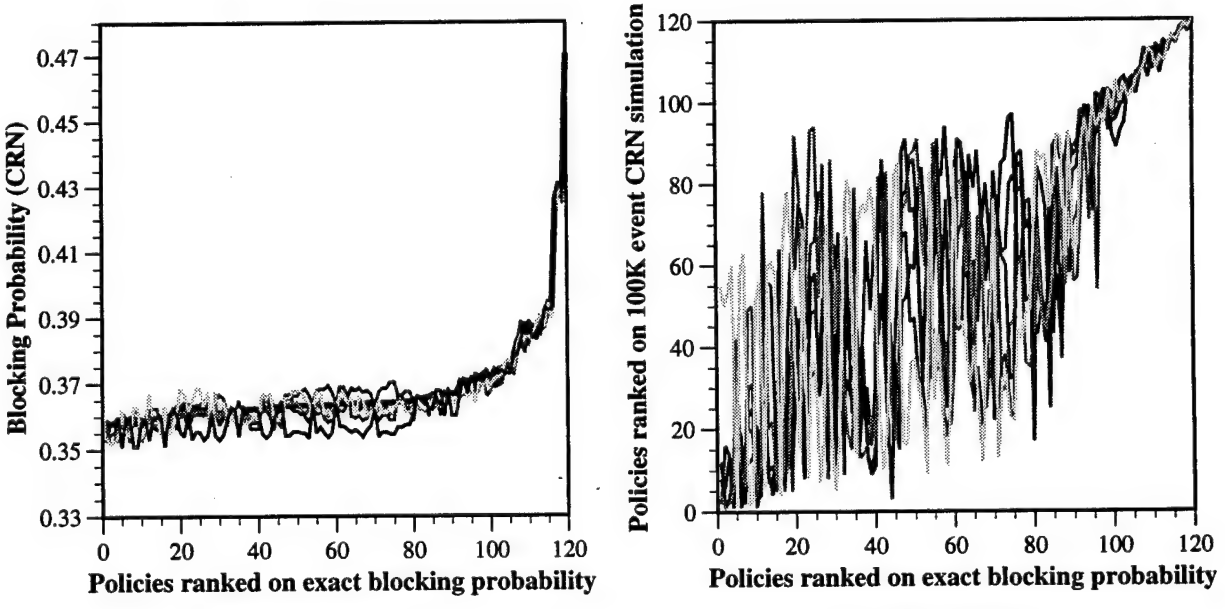
In Section 6 we noted that the use of custom alias tables (custom ratio yardsticks) may, in some cases, permit higher efficiency in SC simulations because of the consequent reduction in the number of fictitious events. However, when custom alias tables are used, different events may be passed to different sample paths, even though the same random-number sequence is used to drive all of the parallel experiments. We now address the quality of the ordinal rankings that are obtained when common random numbers (CRN) and custom alias tables are used to simulate the same example considered above for SC simulation with a common event stream (Section 9.2.2) and for independent BF simulations (Section 9.2.3). In particular, we address the conjecture that the use of a common underlying random-number sequence can introduce beneficial correlation among the sample paths, even though the event sequences are different.

Figures 24 through 26 show the blocking probability and ordinal rankings for simulations of length 10^4, 10^5, and 10^6 events for the present case of CRN simulations. It appears from these curves that little, if any, benefit is achieved as compared to the set of independent experiments of Section 9.2.3. This observation is supported by Fig. 27, which shows the Spearman rank curves for the CRN cases, along with the curves shown earlier in Fig. 23 for the common-event and independent-event examples. (The Spearman rankings for the CRN examples show much greater variation among the different random seeds than those for the common-event and independent-BF examples.) Consequently, the use of custom alias tables in SC simulations appears to be detrimental, because the differences introduced into the experiments reduces the correlation among them, thus resulting in poor ordinal rankings of policies. To obtain comparable ordinal rankings, it may be necessary to run simulations that are several orders of magnitude greater in length than those required for common-event SC simulations. In that case, however, all of the efficiency gained through the reduction of fictitious events would be eliminated.



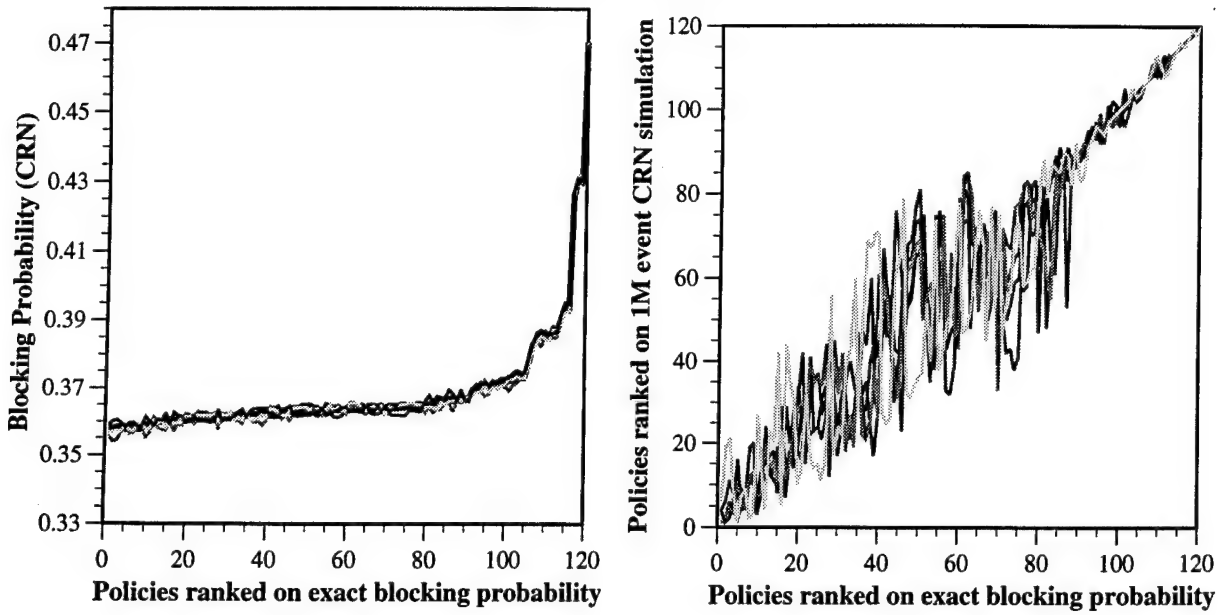
(a) blocking probability (b) ordinal rankings

Fig. 24 – Blocking probability and ordinal rankings for CRN simulations with custom alias tables; six random-number generator seeds; simulation duration = 10^4 events



(a) blocking probability (b) ordinal rankings

Fig. 25 – Blocking probability and ordinal rankings for CRN simulations with custom alias tables; six random-number generator seeds; simulation duration = 10^5 events



(a) blocking probability (b) ordinal rankings

Fig. 26 – Blocking probability and ordinal rankings for CRN simulations with custom alias tables; six random-number generator seeds; simulation duration = 10^6 events

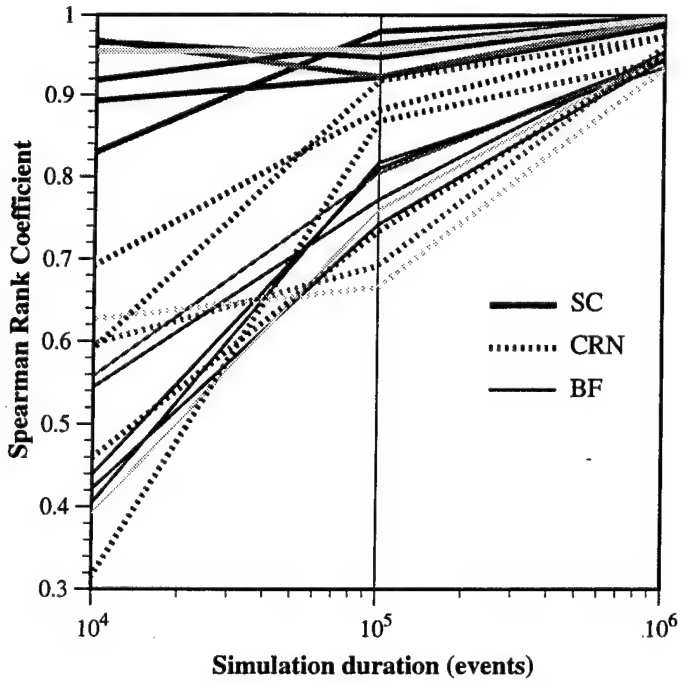


Fig. 27 – Quality of ordinal rankings of blocking probability, based on Spearman's rank coefficient for SC, BF, and CRN simulations of various lengths

10. DATA-PACKET DELAY: PERFORMANCE EVALUATION AND ORDINAL OPTIMIZATION

Now we turn our attention to the simulation of data-packet delay. As for the case of blocking probability discussed in Section 9, we are interested in the accuracy of the simulation-based values of the desired performance measures, as well as the accuracy of the corresponding policy rankings.

10.1 Evaluation of Data-Packet Delay

In Fig. 28 we compare the average data delay found by SC simulations (that processed 10^6 voice arrivals[16]) of different control policies to the value computed by our simple and inaccurate analytical M/D/1 model for delay as discussed in Section 5.3. We again consider the network of Fig. 8a with eight transceivers at each node, and examine the same 120 policies as in the voice-call blocking probability example; however, we plot our results only for the best 87 policies because the estimated delay based on the analytical M/D/1 model is infinite for the remaining policies (the system is in saturation because the offered load is greater than the residual capacity at one or more nodes).

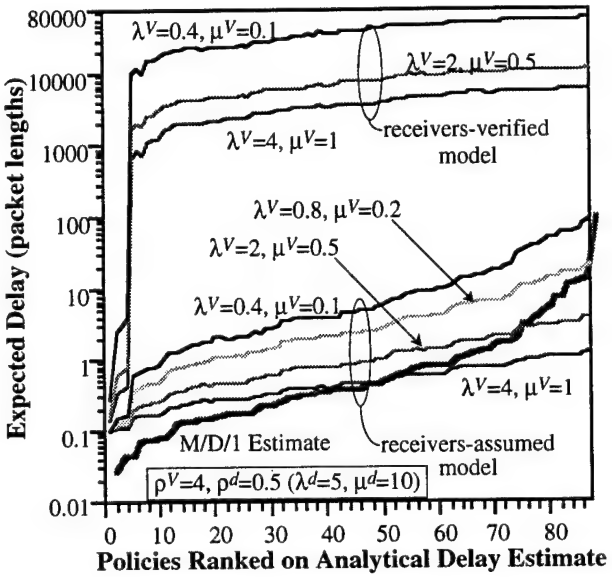


Fig. 28 – Delay measured in SC simulations compared to the analytical estimate

To generate the results shown in Fig. 28, various sets of parameters were used, all of which correspond to $\rho_j^V = \rho^V = 4$ (j = 1, 2, ..., 5) and $\rho_k^d = \rho^d = 0.5$ (k = 1, 2, ..., 10). The extremes in the voice rates were (λ^V= 4.0, μ^V = 1.0) and (λ^V = 0.4, μ^V = 0.1). Since the data service rate μ^d was 10.0 in all the simulations, the expected voice-call duration is 10 times the data-packet length when μ^V = 1.0, and 100 times the data-packet length when μ^V = 0.1. Thus, the total offered voice and data loads are the same in these simulations, but the parameter sets differ in the voice rates λ^V and μ^V. This is of interest because our analytical delay estimate, which is based on the product-form solution of the voice process, only considers the offered voice *load* (i.e., ρ_j^V). It does not account for the actual values of λ^V and μ^V. By keeping the offered load the same, and varying the voice rate (i.e., vary λ^V and μ^V subject to $\lambda^V/\mu^V = 4$), we can evaluate the impact of this simplification in the analytical delay estimate.

In Fig. 28, the x axis represents the control policies ranked in terms of data delay according to our analytical estimate (which we have acknowledged as being inaccurate), with the best policy in position 1. The figure shows the actual delay measured in seven SC simulations. Four of the simulations used the receivers-assumed model, which is similar to the analytical model in that it does not require a transmitter-receiver matching. The other three used the receivers-verified model, in which each transceiver can function as a transmitter or a receiver, but not both simultaneously. As expected, the analytical delay calculation yields a poor estimate of the *actual* delay value. The simulation curves in Fig. 28 show that the slower rates of variation in voice state caused by smaller values of λ^V and μ^V (which correspond to

[16] Based on our experience, a simulation run of this length produces sufficiently accurate delay estimates for the networking examples studied in this report.

longer interarrival times and call durations, and thus in longer periods of heavy voice congestion) result in longer periods in which data packets are forced to remain in queue, and hence in considerably greater delay, even though they all correspond to the same value of ρ. The delay measured under the receivers-verified model is considerably greater than that under the receivers-assumed model because of the requirement to have a transceiver available at both the transmitting and receiving nodes. In fact, for all but the best four policies, the results for the receivers-verified model represent operation in saturation; thus expected queue lengths and delay continue to increase as the simulation progresses.[17]

10.2 Ordinal Ranking of Policies in Terms of Data-Packet Delay

We have investigated three distinct approaches to the ordinal optimization of data-packet delay, namely the use of the approximate M/D/1 queueing model, short simulation runs, and residual bottleneck capacity. We demonstrate in this subsection that each of these methods can provide remarkably accurate (although imperfect) rankings of admission-control policies.

10.2.1 The M/D/1 Queueing Model

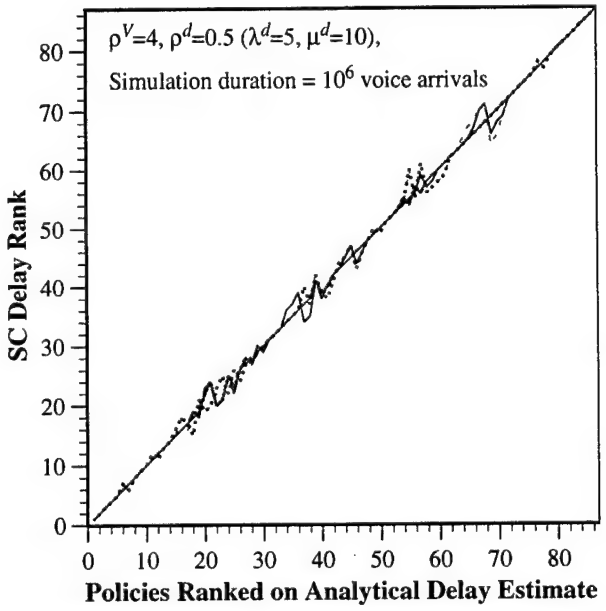
In Fig. 29 we present two views that compare the ordinal rankings obtained from four of the SC simulations of Fig. 28 to those obtained from the simple analytical M/D/1 model for delay. Two of the simulations used the receivers-assumed model and the others used the receivers-verified model. In Fig. 29(a) the y axis represents the rank assigned to the policy (x axis) as a result of SC simulations. Again, a straight line with unit slope would indicate perfect agreement. Figure 29(b) gives a different view of the same data. In this figure we plot the deviation of the delay rankings obtained from SC simulations from the ranking given by the analytical delay estimate (i.e., for policy Ω, Deviation(Ω) = SC rank of Ω – analytical rank of Ω). The figures show that the agreement for all four simulations, although imperfect, is impressively good; the four curves are virtually indistinguishable, and therefore they are not labeled. The deviation in ranking among the top 54 policies is never more than three. Furthermore it is interesting that, despite the significant differences between the simulation models (data receivers assumed vs data receivers verified) and the voice rates ($\lambda^V = 4$ vs $\lambda^V = 0.4$), which produce significantly different values of delay, the rankings are very similar among the different simulation runs. This is true even though, for the case of the receivers-verified model, all but the top four policies represent operation in the saturation region.

10.2.2 Short Simulation Runs

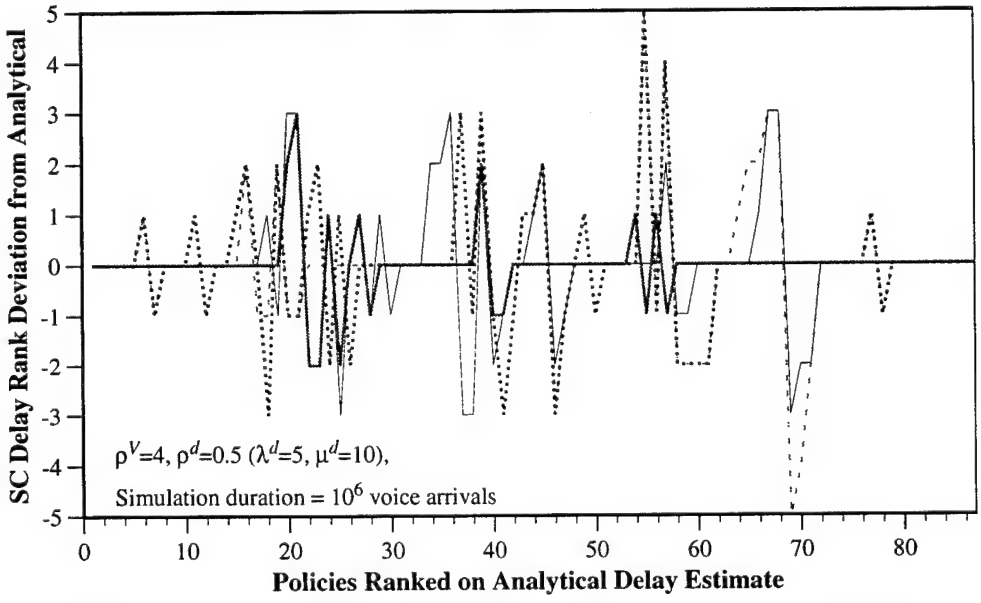
As in the case of voice-call blocking probability, short simulation runs may also be used for ordinal optimization of data-packet delay. Now we express performance in terms of the average deviation (here we use the absolute value of the deviation) of policy rankings for simulations of various lengths for the receivers-verified model, where the "exact" rankings are based on a run of 10^7 events. Figure 30 shows the average delay rank deviation as the simulation length is varied from 10^3 to $5\mathbf{X}10^6$ events. In Fig. 30(a), results are shown for four different random-number seeds. One of the seeds (namely seed 2) provides deviations that are considerably larger than those of the other seeds for short simulation runs; however, the effect of the seed tends to diminish as the length of the simulation grows. For each seed the results are averaged over three different simulation methods based on the discussion of Section 7. In the

[17] Since the number of voice arrivals is the same (i.e., 10^6) for each of the cases shown in Fig. 28, the time duration represented by each simulation run is inversely proportional to λ^V. Once equilibrium is reached (as it is for the cases corresponding to the receivers-assumed model), there is little change in performance. However, when the system is saturated (as it is for the receivers-verified model), the expected delay in long simulation runs is proportional to the duration of the simulation. For example, the duration of the run corresponding to the top curve for the receivers-verified model is ten times that of the bottom curve; the expected delay shown in the top curve is slightly larger than ten times that in the bottom curve.

"direct-processing" method (referred to as DP in the figure), a deterministic-events queue is maintained, whereas in the "improved-processing" (IP) method we take advantage of the constant data-service rate between voice events. Actually, two forms of the IP method are used, one in which the system data state is updated at every stochastic event (IP_1) and one in which updating takes place only at voice events (IP_2). The delay rank deviations for these three methods (in each case averaged over the four random seeds) are shown in Fig. 30(b). It is apparent from this figure that the three simulation methods provide nearly identical rankings. This is significant because the improved-processing methods require approximately an order of magnitude less computation time than the direct-processing method. Moreover, the IP_2 method is significantly more efficient than the IP_1 method because it updates the state only at voice events. Nevertheless, its rankings are virtually indistinguishable from those of the IP_1 method, which updates at all stochastic events.

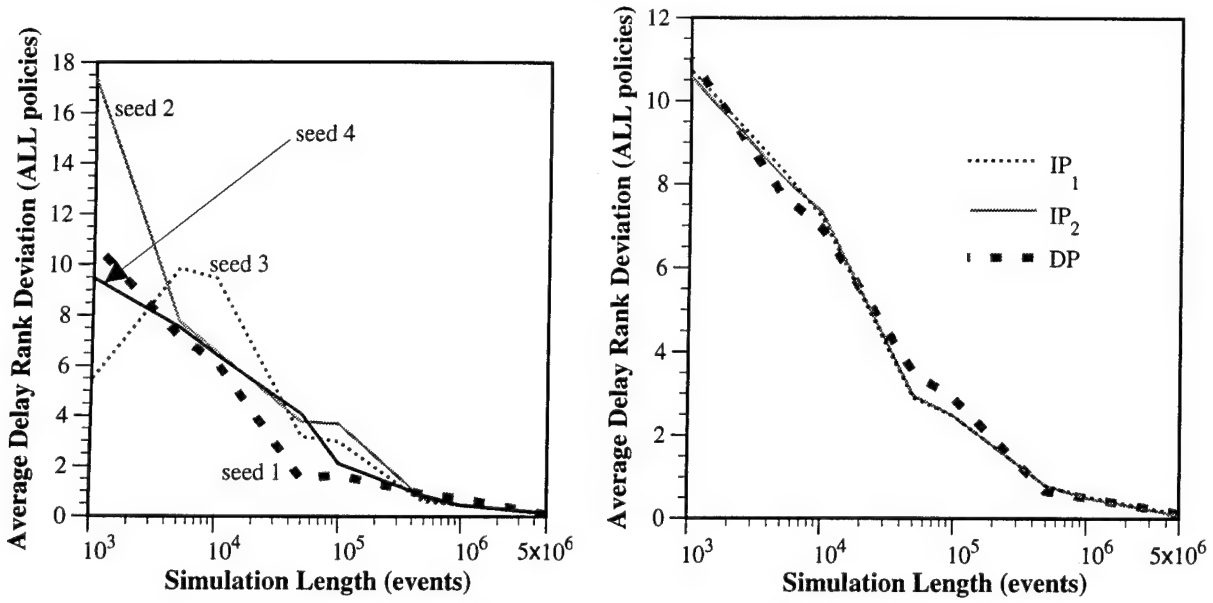


(a) SC delay rank vs analytical delay estimate rank



(b) Deviation of the SC delay rank from the rank given by the analytical delay estimate

Fig. 29 – Comparison of the policy rankings by SC simulation and by the analytical delay estimate



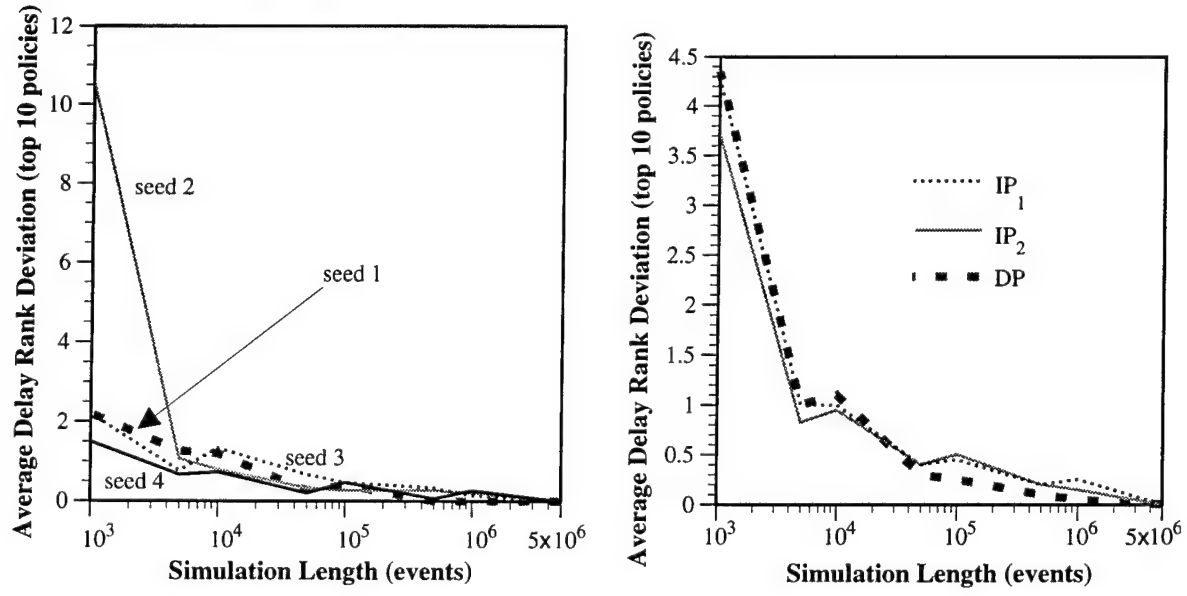
(a) averaged over different simulation models

(b) averaged over different seeds

Fig. 30 – Average delay-rank deviations over the set of all policies; receivers-verified model

Figures 31 and 32 show similar sets of delay-rank deviations for the set of the ten best and four best policies, based on "exact" rankings. These deviations are considerably smaller than those observed in Fig. 30 for the set of all policies. Seed 2 again produces relatively large delay-rank deviations for the short simulation runs.

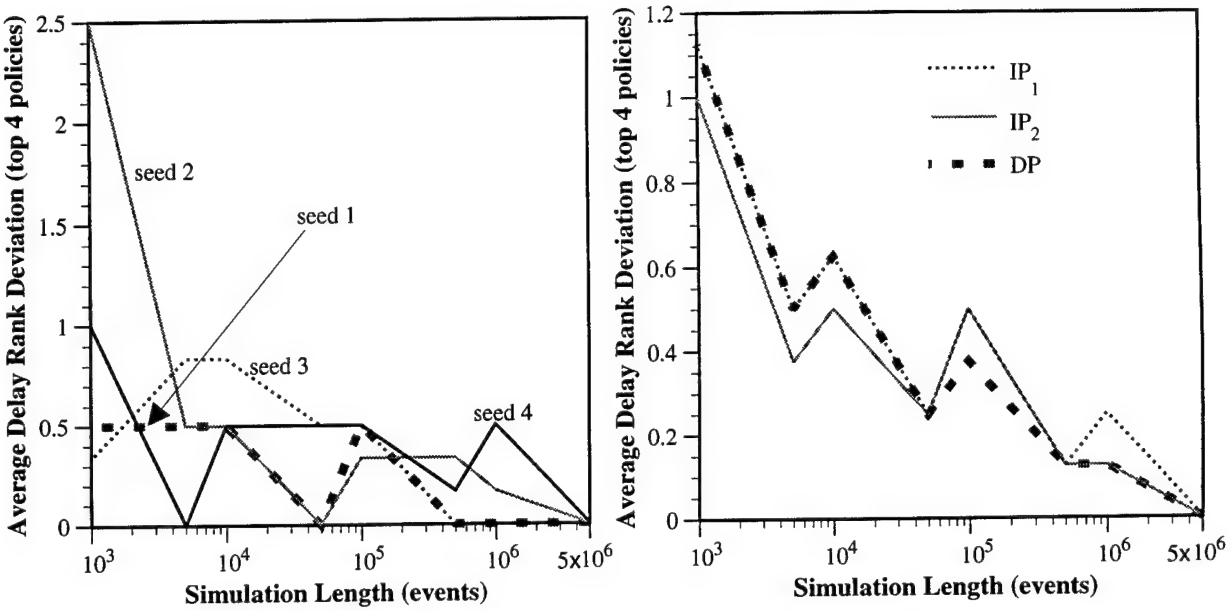
Figure 33, which shows the Spearman rank coefficient vs the simulation duration, demonstrates how the accuracy of the rankings improves as the length of the simulation run increases. This figure shows that the value of r_s is relatively high even for short simulations, and that it rapidly increases to nearly the ideal value of 1.0.



(a) averaged over different simulation models

(b) averaged over different seeds

Fig. 31 – Average delay-rank deviations over the set of the ten best policies, receivers-verified model



(a) averaged over different simulation models

(b) averaged over different seeds

Fig. 32 – Average delay-rank deviations over the set of the four best policies, receivers-verified model

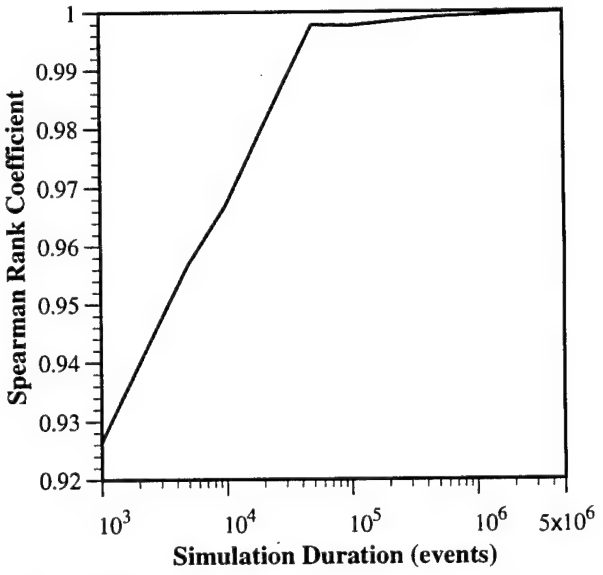


Fig. 33 – Quality of ordinal rankings of data-packet delay, based on Spearman's rank coefficient; receivers-verified model

Now we consider the effect of simulation length on the accuracy of rankings by studying the average deviation of the best n policies from the "true" rankings, which in this case are the rankings after a simulation of 10^6 voice-arrival events duration. In Fig. 34, we present results of this type for the case of the receivers-assumed model. Note that the best 19 policies based on the M/D/1 model are identical to those determined by the long simulation run, and that the average deviation is well under 1 over the entire range of policies. Also note that the best 11 policies were determined in exact order in the simulation of duration 10^4; the average deviation is less than 2 over nearly the entire range of policies. Even the shorter simulation runs do quite well in predicting the rankings for the best 20 policies.

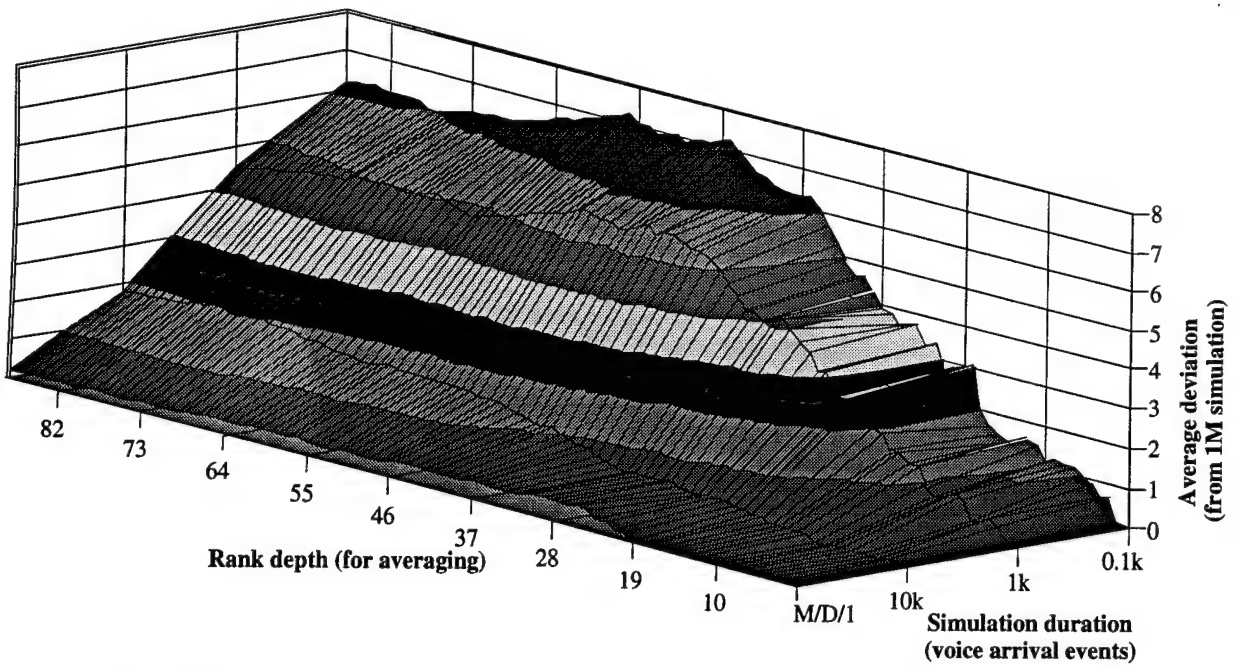


Fig. 34 – Average deviation of policy rankings for the best n policies for simulations of various lengths for the receivers-assumed model and for the M/D/1 model

10.2.3 Residual Bottleneck Capacity

Now we consider ordinal optimization by means of an alternative, and very simple, performance metric. We define the *bottleneck node* to be the node with the minimum value of residual capacity, and the *bottleneck data capacity* C_{bot} to be the residual capacity of this node. Figure 35 compares the ordinal rankings of 120 different policies based on SC simulation data-delay measurements (for $\lambda_i^d = 5$, $\mu_i^d = 10$, therefore $\rho_i^d = 0.5$, $i = 1, \ldots, 10$, and for $\lambda^V = 4$, $\mu^V = 1$, receivers assumed) to ordinal rankings of the same 120 different policies based on C_{bot}. Note that the simulation rankings agree quite well with the residual bottleneck data capacity rankings throughout the set of all policies, with almost perfect agreement in the first 15 positions. However, despite the high quality of the ordinal rankings obtained in this example, we expect that the usefulness of the metric C_{bot} may be highly problem-dependent. This is because this metric does not take into account the data-traffic statistics. For example, a traffic loading that puts a very light data-traffic load through the bottleneck node but a heavy data traffic load through other nodes can result in poor ordinal rankings. We expect that C_{bot} will tend to perform well under relatively uniform data-traffic loads.

11. OBSERVATIONS ON THE USE OF ORDINAL OPTIMIZATION

The results of this section provide evidence of the potential value of ordinal-optimization methods. When we are interested in optimizing performance, the ordinal ranking of the different policies (i.e., the ranking from best to worst) is much more informative than the actual measure of performance under a given policy. In our examples here, Figs. 28 and 29 show that although our analytical delay estimate is inaccurate, it does very well in selecting the best control policies with respect to data-packet delay. For example, although the time constants associated with voice (λ_j^V and μ_j^V) have a major impact on data-packet delay, they only minimally affect the policy ranking. Also, we have observed that the relative rankings of policies are quite insensitive to the requirement (or lack of it) of having a receiver available for each data-packet transmission. We feel that it is especially significant that simple analytical models, such as the M/D/1 queue and even the residual bottleneck data capacity, can provide highly accurate rankings.

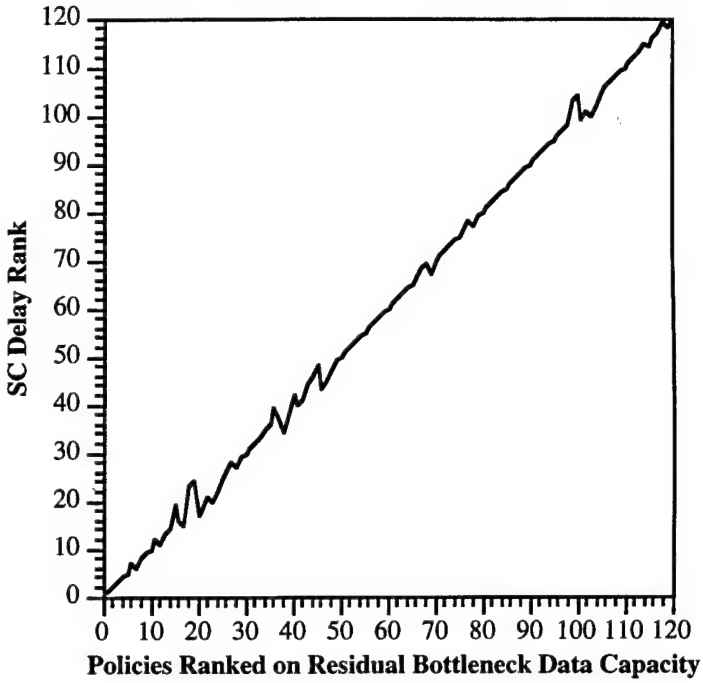


Fig. 35 – Comparison of policy rankings based on residual bottleneck data capacity and the data delay measured in SC simulations

The remarkable similarity of the delay rankings obtained by using the different models may have interesting, and possibly far-reaching, implications in the study of optimization methods. These results suggest that accurate ordinal rankings can be obtained even when highly inaccurate models (in the sense of predicting actual performance values) are used. The principal requirement here appears to be the identification of the key aspects of the model that affect relative performance. In our examples the residual capacity for data is of primary importance, while the time constants associated with voice traffic and the availability of receivers at destination nodes are relatively unimportant. Thus, simulation under one set of parameters provides a good indication of relative performance under other parameter sets (for the same values of utilization ρ_j^v).[18] Moreover, the accuracy of the rankings obtained by using the analytical approximation suggests that simulation may actually be unnecessary to determine the best policy or best set of policies. However, simulation will still be necessary to evaluate the system performance under these policies.

To obtain an intuitive explanation of why crude models can provide accurate rankings, we may view each link as a service system. In general, delay in such systems is a convex increasing function of system load. Figure 36 shows plots of delay vs load for an M/D/1 model and for the true service system (a selected link) with the same average capacity. Although the value of delay in the two systems is quite different, the delay curves are both convex increasing functions that approach infinity as the load approaches system capacity. Thus a policy that performs well for the M/D/1 system should perform well for any "similar" system. For example, use of policy Ω_i determines the expected residual capacity at the link of interest,[19] which is the same for both systems (the true service system and the M/D/1 model). The residual capacity, in turn, determines the expected data service rate μ^d, and hence the expected data load $\rho^d(\Omega_i) = \lambda^d/\mu^d$. Thus each policy Ω_i results in a corresponding load $\rho^d(\Omega_i)$ at the link of interest, which is

[18] We have observed that this is true provided that the expected voice-call duration is greater than or equal to ten times the data-packet length.

[19] Actually at all links in the network.

the same for both systems. Because of the convexity of the performance measures of both systems (as a function of load), the ordinal rankings of policies are the same for both systems.

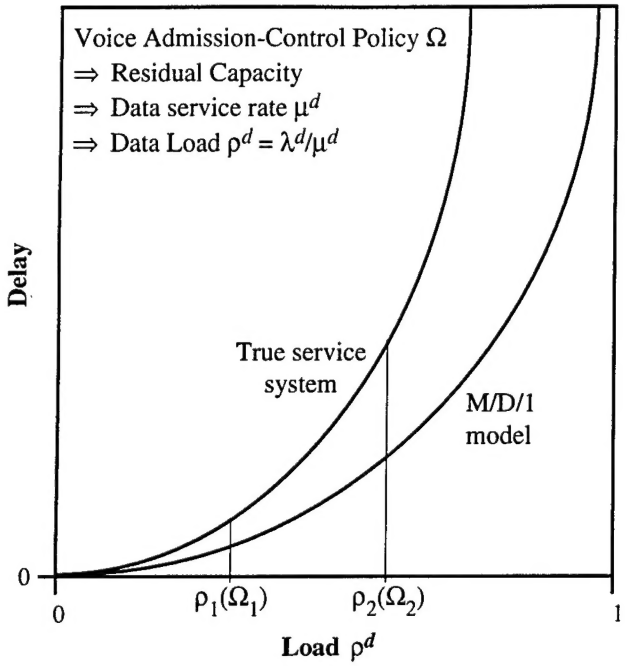


Fig. 36 – Delay vs load for an M/D/1 queueing system and "the true" service system

Actually, the performance measure of interest on a global (network-wide) basis is the average delay, where the average is taken over all queues in the system. Since the expected delay at each node is a convex function of the load on that node, the overall expected network delay averaged over all nodes is the convex combination of a number of convex functions, resulting in an overall convex and increasing function of the combined total load (i.e., the multidimensional load vector representing the traffic at all nodes). Thus the use of a simplified model at each queue in the network should provide an indication of global performance as well. However, the situation is complicated by the fact that the ordinal rankings of policies are not necessarily the same at all queues in the network. For example, whereas Ω_1 may provide a greater residual capacity than Ω_2 at one queue, the opposite may be true at another. This is one reason that the ordinal rankings obtained by means of crude models are not perfect.

12. SUMMARY AND CONCLUSIONS

In this report we have studied several methods to improve the efficiency of the simulation of integrated networks for the purpose of performance evaluation and, more importantly, for the purpose of optimization. Our primary approach is the use of the standard clock (SC) technique [1-3], which is an approach for the parallel simulation of structurally similar DEDS that operate under different parameter values or control policies. First, in studies of M/M/1/K systems we quantified the reduction in simulation time that can be achieved when the SC approach is used on a sequential machine, and demonstrated that there is a limit on the speedup that can be achieved as the number of parallel experiments increases. We then turned our attention to circuit-switched wireless networks and demonstrated that similar improvements can be achieved there as well. We have extended the applicability of the SC approach (a method directly applicable to systems with exponential interarrival times) to the case of integrated networks with fixed-length data packets, and have demonstrated that this is an efficient method for the parallel simulation of networks of this type operating under a number of different control policies.

Often, it is more important to find the optimal control policy than to quantify the performance achieved by implementing that policy. In practice it is more realistic to search for a good policy that performs almost as well as the optimal one, which may remain elusive because of the vast search space that has to be explored. We have demonstrated that it is possible to achieve a remarkably accurate ranking of the relative performance of the control policies relatively early in the simulation, thus permitting the efficient determination of good, although not necessarily optimal, policies. The ability to rapidly determine good ordinal rankings is a consequence of the use of a common event sequence to drive all of the parallel simulations.

Furthermore, our results suggest that accurate ordinal policy rankings can be obtained even when highly inaccurate simulation models or analytical approximations are used. We feel that it is especially significant that simple analytical models, such as the M/D/1 queue and the residual bottleneck data capacity, can provide highly accurate rankings on the basis of delay in an integrated network. The agreement of the delay rankings obtained using significantly different simulation models and analytical delay estimators indicates that the use of a simple model, which incorporates the system's key aspects, may suffice to accurately rank the relative performance under different parameters and control policies. The accuracy of the rankings produced by the analytical approximation suggests that, in some applications, simulation may not even be needed to determine good control policies. After the set of all control policies is narrowed to a set of good control policies by means of the analytical approximation, their performance can be evaluated by simulation to determine the best policy.

In conclusion, we have demonstrated that SC techniques and a variety of ordinal optimization methods are effective in speeding up the evaluation/optimization process for an important class of communication network problems. Furthermore, these two approaches work together synergistically to provide a means for the rapid determination of good control policies. Clearly, further research is needed to verify the effectiveness of ordinal-optimization methods in more-general scenarios. In addition to the study of different network topologies, it would be desirable to investigate whether admission-control policies that work well for single-hop data traffic (the case studied in this report) also work well for the more-general case of multihop data traffic. However, the results obtained thus far suggest that ordinal-optimization techniques based on simple analytical models would be easy to implement in practical communication systems. Thus, this approach is expected to provide simple, but nearly optimal, solutions to highly complicated networking problems for which exact solutions are either unavailable or impractical to implement.

REFERENCES

1. P. Vakili, "Using a Standard Clock Technique for Efficient Simulation," *Operations Research Letters*, **10,** 445-452 (1991).
2. Y.-C. Ho, S. Li, and P. Vakili, "On the Efficient Generation of Discrete Event Sample Paths under Different Parameter Values," *Mathematics and Computation in Simulation*, **30,** 347-370 (1988).
3. C. G. Cassandras, J.-I. Lee, and Y.-C. Ho, "Efficient Parametric Analysis of Performance Measures for Communication Networks," *IEEE Journal on Selected Areas in Communications*, **8**(9), 1709-1722 (1990).
4. Y.-C. Ho, R. S. Sreenivas, and P. Vakili, "Ordinal Optimization of DEDS," *Journal of Discrete Event Dynamic Systems*, **2**, 61-88 (1992).
5. J. M. Aein, "A Multi-User-Class, Blocked-Calls-Cleared, Demand Access Model," *IEEE Transactions on Communications*, **COM-26**(3), 378-385 (1978).
6. L. Kleinrock, *Queueing Systems Volume 1: Theory* (John Wiley & Sons, New York, 1975).
7. H. C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach* (John Wiley & Sons, Chichester, 1986).

8. P. Bratley, B. L. Fox, and L. E. Schrage, *A Guide to Simulation* (Springer Verlag, New York, 1987).
9. Y.-C. Ho, C. G. Cassandras, and M. Makhlouf, "Parallel Simulation of Real Time Systems via the Standard Clock Approach," *Mathematics and Computers in Simulation*, **35,** 33-41 (1993).
10. C.-H. Chen and Y.-C. Ho, "Extension of the Standard Clock Method for Discrete Event Simulation," submitted to *IEEE Transactions on Control Systems Technology* (1993).
11. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "Efficient Simulation of DEDS by Means of Standard Clock Techniques: Queueing and Integrated Radio Network Examples," NRL Report NRL/MR/5521--93-7392, September 1993.
12. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Use of the Standard Clock to Improve Simulation Efficiency: A Quantitative Study Based on the M/M/1/K Queue," Proceedings of the 27th Conference on Information Sciences and Systems, Baltimore, MD, March 1993, pp. 112-118.
13. C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis* (R. D. Irwin, Inc. and Aksen Associates, Inc., Homewood, IL, 1993).
14. Sun Microsystems, Inc., *C Programmer's Guide*, 1988.
15. J.-C. Liu and S. M. Shahrier, "On Predictability of Caches for Real-Time Applications," Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'94), Durham, NC, Jan.-Feb. 1994, pp. 52-56.
16. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "An Approach to Voice Admission Control in Multihop Wireless Networks," Proceedings of IEEE INFOCOM'93, San Francisco, CA, March 1993, pp. 246-255.
17. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Admission Control in Integrated Voice/Data Multihop Radio Networks," NRL Report NRL/MR/5521--93-7196, January 18, 1993.
18. G. J. Foschini and B. Gopinath, "Sharing Memory Optimally," *IEEE Transactions on Communications*, **COM-31** (3), 352-360 (1983).
19. S. Jordan and P. Varaiya, "Control of Multiple Service, Multiple Resource Communication Networks," *IEEE Transactions on Communications*, **42**(11), 2979-2988 (1994).
20. S. Jordan and P. Varaiya, "Throughput in Multiple Service, Multiple Resource Communication Networks," *IEEE Transactions on Communications*, **39**(8), 1216-1222 (1991).
21. D. Y. Burman, J. P. Lehoczky, and Y. Lim, "Insensitivity of Blocking Probabilities in a Circuit-Switching Network," *Journal of Applied Probability*, **21**, 850-859 (1984).
22. G. J. Coviello and P. A. Vena, "Integration of Circuit/Packet Switching by a SENET (Slotted Envelope Network) Concept," Record of National Telecommunications Conference, 1975, pp. 42.12-42.17.
23. J. E. Wieselthier and A. Ephremides, "Fixed- and Movable-Boundary Channel-Access Schemes for Integrated Voice/Data Networks," *IEEE Transactions on Communications*, **43**, 64-74 (1995).
24. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Improvement in Simulation Efficiency by Means of the Standard Clock: A Quantitative Study," Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio TX, December 1993, pp. 2217-2223.
25. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "Ordinal Optimization of Admission Control in Wireless Multihop Voice/Data Networks via Standard Clock Simulation," Proceedings of IEEE INFOCOM'94, Toronto, Ontario, Canada, June 1994, pp. 29-38.
26. H. A. David, *Order Statistics, Second Edition* (Wiley, New York, 1982).
27. Y.-C. Ho and M. E. Larson, "Ordinal Optimization Approach to Rare Event Probability Problems," submitted to *Journal of Discrete Event Dynamic Systems* (1993).

28. W.-G. Li, N. T. Patsis, and Y.-C. Ho, "Performance Analysis of Scheduling Algorithms in a Gigabit ATM Switch using Ordinal Optimization," submitted to *IEEE/ACM Transactions on Networking* (1993).
29. N. T. Patsis, C.-H. Chen, and M. E. Larson, "SIMD Parallel Discrete Event Dynamic System Simulation," submitted to *IEEE Transactions on Control Systems Technology* (1993).
30. S. Kotz and N. L. Johnson, ed., *Encyclopedia of Statistical Sciences* (Wiley, New York, 1982), pp. 584-587.
31. P. Glasserman and P. Vakili, "Comparing Markov Chains Simulated in Parallel," *Probability in the Engineering and Informational Sciences*, **8**(3) (1994).
32. P. Glasserman and D. D. Yao, "Some Guidelines and Guarantees for Common Random Numbers," *Management Science*, **38**(6), 884-908 (1992).
33. P. Heidelberger and D. I. Iglehart, "Comparing Stochastic Systems Using Regenerative Simulation with Common Random Numbers," *Advances in Applied Probability*, **11**, 804-819 (1979).